SOMMARIO

Packet filtering con Linux

IPChains (Linux 2.2.x)

NetFilter (Linux 2.4.x/2.6.x)

INTRODUZIONE

Con lo sviluppo del kernel Linux, una delle componenti che è stata sottoposta via via ad una quasi totale riscrittura è il supporto al networking, soprattutto per quanto riguarda la gestione dei pacchetti di rete.

Se si esamina la gestione di un firewall Linux, dal primo rudimentale sistema di packet filtering e firewalling implementato nel kernel 1.2.8 (basato sostanzialmente sul modello logico del NetBSD), al complesso sistema di netfilter che disponibile col kernel 2.4.0, pare di trovarsi di fronte a due sistemi operativi completamente differenti.

Il codice di gestione e filtraggio dei pacchetti originale aveva delle limitazioni notevoli. I frammenti non venivano gestiti, i counters erano solo a 32 bit, non si potevano specificare regole per protocolli diversi da TCP, UDP, ICMP. Oltre a questo, non erano possibili riconfigurazioni al volo delle regole, il che rendeva difficile, se non impossibile, l'impiego di Linux come firewall in ambienti dove la sicurezza è un requisito fondamentale. È quindi meglio tralasciare versioni del kernel ormai superate technicamente ed esaminare la gestione dell'ipchains propria dei Linux 2.2 e del netfilter dei Linux 2.4 e 2.6.

Bisogna però avvertire che questo non è un semplice manuale implementativo: ce ne sono di ottimi, anche pubblicamente disponibili, e non avrebbe senso proporne un ennesimo.

Naturalmente verrà descritto come creare un firewall e come amministrarlo, e verranno presentati tutti i comandi di sistema con le loro opzioni e file di configurazione.

Tuttavia non ci si rifarà allo schema applicativo di una singola distribuzione, ma alla riga di comando disponibile su tutte le distribuzioni Linux, su qualsiasi piattaforma hardware.

Un particolare approfondimeto sarà inoltre destinato alla descrizione del modello teorico comune che è sotteso a queste due implementazioni, ed alle altre caratteristiche della gestione del networking a livello kernel che, anche se non strettamente correlate col firewalling, consentono di aumentare le prestazioni e la sicurezza del firewall stesso.

Packet Filtering con Linux

Tutto il traffico che passa attraverso una rete viaggia sotto forma di pacchetti. Più pacchetti concatenati vengono a costituire un messaggio complesso. La taglia dei pacchetti, naturalmente, varia a seconda del protocollo di rete, a seconda della MTU *(maximum transfer unit)* e della MRU *(maximun receive unit)* che sono state configurate sull'interfaccia di rete.

I pacchetti vengono poi ricomposti in unità significative tramite la concatenazione dei serial e degli ack. L'inizio di ogni pacchetto riporta l'indirizzo di destinazione, quello di partenza, il protocollo di comunicazione ed il checksum.

Oltre a questo sono impostati quattro bit, che costituiscono il TOS *(Type on Service)*; essi impostano il Minimum Delay, il Maximum Throughput, la Maximum Reliability ed il Minimum Cost per la trasmissione dei dati contenuti. Solo uno di questi bit può essere settato in ogni pacchetto. Tale insieme di informazioni si chiama header, la parte restante del pacchetto prende il nome di body (vedi *figura 1*).

L'UDP *(User Datagram Protocol)* è un protocollo senza connessione, quando un pacchetto viene trasmesso, il mandante non si cura se esso sia stato ricevuto o meno.

Il TCP *(Transmission Control Protocol)* deve creare una connessione: prima di qualsiasi trasmissione di dati avviene uno scambio di pacchetti i cui header indicano che si tratta di pacchetti di setup, il cui contenuto è pressapoco:

```
- I want to connect
- OK
- Thanks
```

tale procedura prende il nome di “hand shake”.

Inoltre i pacchetti TCP sono numerati, ed in base alla sequela di serial ed ack i due punti della comunicazione si accertano che essa sia stata effettuata correttamente. I pacchetti di risposta ad una richiesta di connessione si chiamano **SYN** packets, in quanto hanno settato il *SYN flag*, mente il **FIN** e l’**ACK** sono vuoti. Se queste risposta non viene ricevuta la connessione non può aver luogo.

Nell’header oltre al protocollo vero e proprio viene anche indicata la porta di destinazione, vale a dire il canale che deve essere utilizzato per la comunicazione. Alcuni canali sono riservati a comunicazioni di dati specifiche.

Nel file **/etc/services** in qualsiasi sistema Unix sono mappate tutte le porte riservate, del quale vediamo un breve esempio qua di seguito:

```
rtmp            1/ddp    #Routing Table Maintenance Protocol
tcpmux          1/tcp    #TCP Port Service Multiplexer
tcpmux          1/udp    #TCP Port Service Multiplexer
nbp             2/ddp    #Name Binding Protocol
compressnet     2/tcp    #Management Utility
```
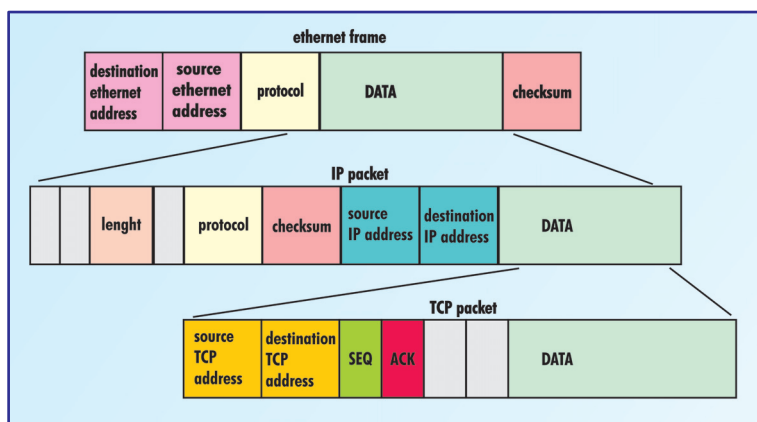
Figura 1: TCP/IP Protocol Layer

```
compressnet      2/udp    #Management Utility
compressnet      3/tcp    #Compression Process
compressnet      3/udp    #Compression Process
echo             4/ddp    #AppleTalk Echo Protocol
rje              5/tcp    #Remote Job Entry
rje              5/udp    #Remote Job Entry
zip              6/ddp    #Zone Information Protocol
echo             7/tcp
echo             7/udp
discard          9/tcp    sink null
discard          9/udp    sink null
systat           11/tcp   users             #Active Users
systat           11/udp   users             #Active Users
daytime          13/tcp
daytime          13/udp
qotd             17/tcp   quote             #Quote of the Day
qotd             17/udp   quote             #Quote of the Day
msp              18/tcp   #Message Send Protocol
msp              18/udp   #Message Send Protocol
chargen          19/tcp   ttytst source     #Character Generator
chargen          19/udp   ttytst source     #Character Generator
ftp-data         20/tcp   #File Transfer [Default Data]
ftp-data         20/udp   #File Transfer [Default Data]
ftp              21/tcp   #File Transfer [Control]
ftp              21/udp   #File Transfer [Control]
ssh              22/tcp   #Secure Shell Login
ssh              22/udp   #Secure Shell Login
telnet           23/tcp
telnet           23/udp
smtp             25/tcp   mail              #Simple Mail Transfer
smtp             25/udp   mail              #Simple Mail Transfer
nsw-fe           27/tcp   #NSW User System FE
nsw-fe           27/udp   #NSW User System FE
msg-icp          29/tcp   #MSG ICP
msg-icp          29/udp   #MSG ICP
msg-auth         31/tcp   #MSG Authentication
msg-auth         31/udp   #MSG Authentication
dsp              33/tcp   #Display Support Protocol
dsp              33/udp   #Display Support Protocol
time             37/tcp   timserver
time             37/udp   timserver
```

Altre porte, come quelle da 1019 a 65535 sono riservate alla creazione dei

socket di comunicazione dopo che la connessione è stata effettuata.

Tali porte vengono quindi dette “non privilegiate”. In realtà le porte non privilegiate partono dalla numero 1024, ma parecchi software, come ad esempio ssh standard 1, usano bensì le porte comprese da 1019 a 1023 per le connessioni invece delle “non privilegiate” tradizionali.

Si tratta di un dato fondamentale, che tutti i network manager dovrebbero tenere ben presente, poiché, come sarà chiaro in seguito, spesso obbliga a ripensare in toto le regole sulla base delle quali si filtra il traffico su parecchie reti.

Il protocollo ICMP non ha porte. I pacchetti ICMP si destinguono in base a *type* e *code*. Il type viene indicato tramite notazione numerica. Questa è una piccola tabella coi più comuni ICMP types

Numero	Nome	Richiesto da
0	echo-reply	ping
3	destination-unreachable	Traffico TCP/UDP
5	redirect	routing se non si usa un routing daemon
8	echo-request	ping
11	time-exceeded	traceroute

Quando un pacchetto è troppo grande il relazione all’MTU ed all’MRU esso viene spezzato in più frammenti inviati come pacchetti multipli il cui header consente la ricostruzione del pacchetto unitario.

Tecnicamente la gestione dell’invio dei frammenti è piuttosto facile, complessa invece è quella della ricezione, in quanto l’ordine di arrivo a destinazione è arbitrario, per cui la bufferizzazione dei dati ricevuti diviene difficoltosa.

Gli “ICMP Fragmentation Needed” packet servono per l’appunto a gestire la frammetazione dei pacchetti. Può capitare che alcuni ISP, definiti provocatoriamente dai Linux kernel hacker come *“criminally braindead”*, blocchino tutto il traffico *icmp* fra la loro intranet ed internet senza operare alcuna distinzione in base al type.

Di conseguenza se l’MSS dei TCP syn packet, ossia i pacchetti di risposta ad una connessione, sono di dimensioni maggiori della MTU, la comunicazione risulta impossibilitata. Solitamente la dimensione dell’MSS deve essere inferiore all’MTU di 40 byte.

Qualora così non fosse ci sono vari artifizi per mutare la situazione. In base all’header dei pacchetti è possibile operare un filtro su di essi, ossia stabilire cosa

debba fare il ricevente nei confronti dei dati inviatigli, se debba accettarli *(accept)*, rifiutarli *(reject)*, o addirittura scartarli come se non li avesse mai ricevuti *(deny)*. Questo tipo di operazione si chiama packet filtering ed il trattamento dei dati prende il nome di politica di filtraggio.

In linea di principio il filtering è operabile su qualsiasi macchina connessa alla rete. Per esempio, si potrebbe volere che una macchina agisca da web server, ma che non accetti qualsiasi altro tipo di pacchetto o che li ridiriga altrove. Si tratta di una soluzione estrema e poco flessibile; la gestione dei vari servizi di rete su
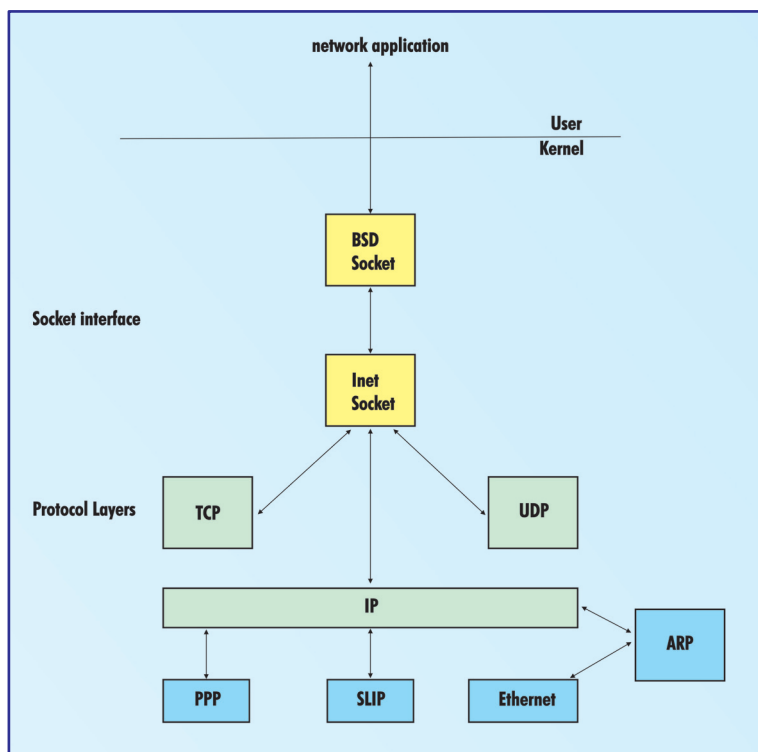
Figura 2: Linux Network Layer
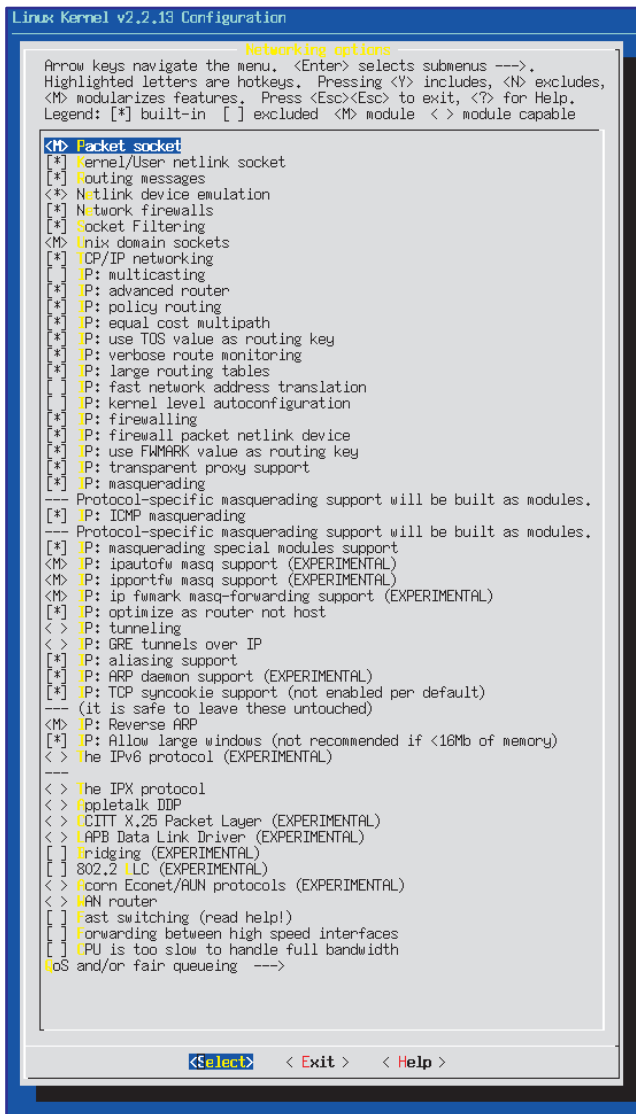
Figura 3: Configurazione di ipchains con un kernel 2.2.13

singole macchine va effettuata con metodologie differenti.

Solitamente il filtering viene operato su entità che fungono da gateway fra diverse intranet o fra delle intranet e delle extranet, di modo tale da regolare il flusso ed il tipo di dati che viaggiano fra le varie reti. Questo tipo di entità viene chiamato comunemente firewall.

Il packet filter è implementabile secondo tre schemi differenti:

- a livello hardware con una interfaccia software (Cisco);
- nei sistemi a microkernel, completamente a livello software (Microsoft Windows NT)
- nei sistemi a macrokernel o a kernel modulare, come built-in del networking layer del kernel stesso (Linux) .

Sotto Linux la politica di filtering dei pacchetti è direttamente registrata all'interno dell'area dati in cui si è mappato il kernel, all'interno del network layer. Si tratta di una soluzione che consente di operare con la stessa efficienza di un filtro hardware, ma con una maggiore flessibilità dal punto di vista amministrativo *(figura 2)*.

Una visualizzazione immediata del network layer può essere ottenuta osservando il contenuto dei file di report nel filesystem proc (familiarmente il "procfs"). Si tratta di un filesystem virtuale che, oltre a registrare lo stack dei processi in corso e tutte le informazioni ad esso irrelate, contiene delle directory con file immagini dello stato attuale dei Layer. Cambiando il contenuto di alcuni di questi file si può anche mutare al volo la configurazione del kernel stesso.

In **/proc/net** sono contenuti i file di reportistica del network layer.

File	Contenuto
Arp	Kernel ARP table
Dev	network devices with statistics
dev_mcast	the Layer2 multicast groups a device is listening too (interface index, label, number of references, number of bound addresses).
dev_stat	network device status
ip_fwchains	Firewall chain linkage
ip_fwnames	Firewall chain names
ip_masq	Directory containing the masquerading tables (2.2)
ip_masquerade	Major masquerading table (2.2)
netstat	Network statistics
raw	raw device statistics

route	Kernel routing table
rpc	Directory containing rpc info
rt_cache	Routing cache
rt_cache_stat	Routing cache statistics (2.4)
snmp	SNMP data
sockstat	Socket statistics
tcp	TCP sockets
tr_rif	Token ring RIF routing table
udp	UDP sockets
unix	UNIX domain sockets
wireless	Wireless interface data (Wavelan etc)
igmp	IP multicast addresses, which this host joined
psched	Global packet scheduler parameters.
Netlink	List of PF_NETLINK sockets
ip_mr_vifs	List of multicast virtual interfaces
ip_mr_cache	List of multicast routing cache

In **/proc/sys/net/** stanno invece i file di riconfigurazione "on the fly".

1.1 Configurazione del kernel

Solitamente il kernel generico fornito dalle distribuzioni per il primo boot del sistema non è configurato per contenere le funzionalità del firewall. È consigliabile compilare nel kernel anche il supporto per i SYN cookies, per essere difesi contro attacchi con SYN flooding, ed il supporto per il routing da kernel.

A seconda del dimensionamento della macchina si deve decidere come configurare il routing del kernel.

In fase di configurazione bisogna decidere se si desidera poter mascherare i pacchetti o meno. Ciò ha rilevanza esclusivamente per i firewall veri e propri. Ci si può infatti limitare ad un filtraggio semplice, oppure i pacchetti possono essere mascherati, ossia il loro header venire modificato, di modo che assumano come source address quello dell'interfaccia di rete da cui vengono rilasciati da parte del firewall/router, al posto di quello del client da cui sono stati effettivamente emessi.

Il processo inverso deve naturalmente essere applicato ai pacchetti di risposta, i quali arrivano con destination address corrispondente all'interfaccia di rete esterna del firewall, mentre è necessario che vengano routati e manipolati in modo che il destination address punti poi al client interno corretto. Questa

operazione si chiama demascheratura.

L'algoritmo di scheduling con cui il kernel gestisce i pacchetti è normalmente il *fifo*, ma se si hanno diverse intranet, con diverse bande passanti, ed un link sulla extranet non proprio performante, è possibile compilare come moduli anche altri algoritmi di schedulazione e dei classificatori di pacchetti, da caricare in relazione ad una data interfaccia di rete a seconda delle necessità contingenti.
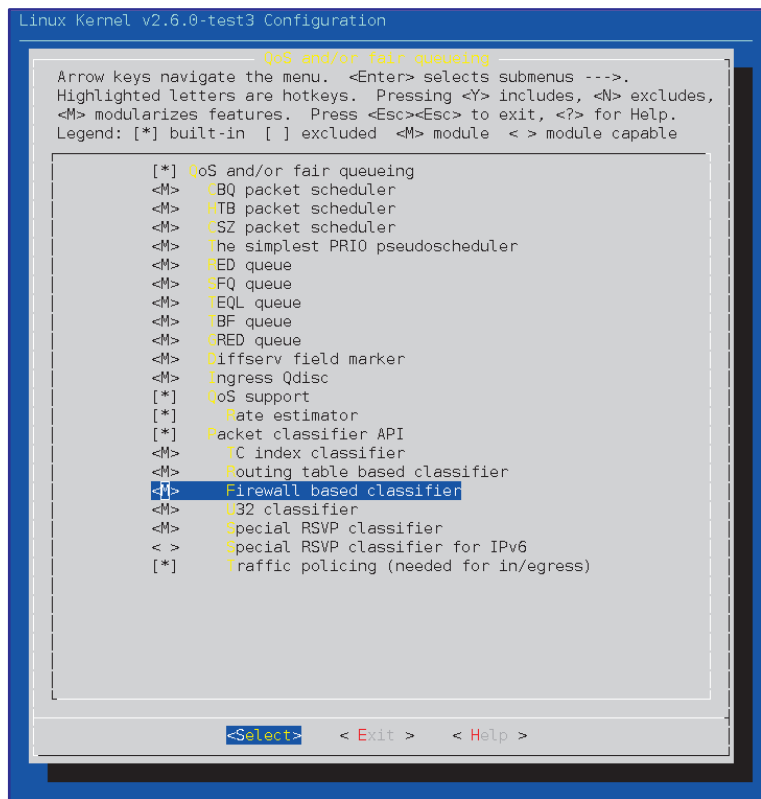
Figura 4: configurazione del QoS con un kernel 2.6

1.2 Principi basilari di funzionalità del packet filter

In questo modo è possibile configurare il così detto Quality of Services (QoS), ed implementare un avanzatissimo traffic shaper, col quale si può sezionare a piacimento la banda passante a secondo del tipo di pacchetti, dei servizi che si intendono privilegiare, ed anche degli host con cui si comunica.

Per spiegare nel dettaglio il complesso processo del filtraggio è bene prenderne la forma più semplice, ossia quella di ipchains propria dei kernel 2.2. In effetti la metodologia con cui viene applicato il filtraggio nei kernel 2.4 e 2.6 (chiamato indifferentemente netfilter od iptables) è sottilmente diversa.

Ciò condiziona la formulazione ed il posizionamento delle regole all'interno dei chain, e di conseguenza il modo di concepire il firewall stesso. Per evitare inutili confusioni, e qualche travisamento metodologico, sarà però oppurtuno soffermarsi sul funzionamento dettagliato dell'avanzatissimo Netfilter di linux 2.4 molto più avanti, quando il lettore avrà ormai preso pratica e familiarità con ipchains, e sarà quindi dotato degli strumenti conoscitivi necessari per penetrare nei meandri della complessa logica di iptables.

Il kernel di per sé comprende solo tre liste di regole di filtraggio dei pacchetti, dette filter chain; esse sono: **input**, **output** e **forward**.

Quando un pacchetto arriva, viene applicato il chain dell'input per stabilire se debba essere accettato. In caso sia data risposta affermativa, allora il kernel decide
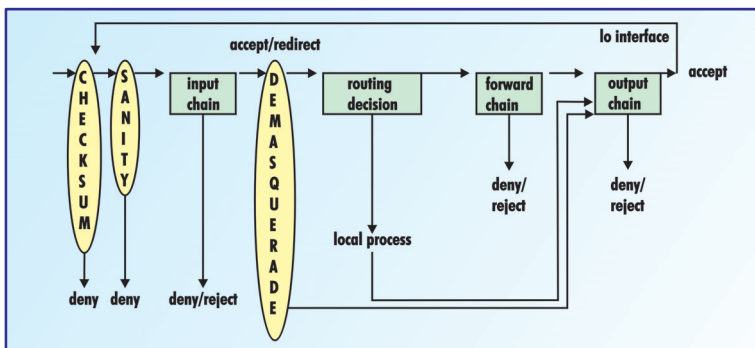
Figura 5: Netfiltering Layer

dove il pacchetto debba essere inviato (questo è il processo di routing). Qualora la destinazione sia un'entità differente al firewall stesso viene effettuato il forward, e prima che il pacchetto esca il kernel consulta il chain di output.

A meno che il pacchetto non venga bloccato da una regola che in base al tipo di header decida per il drop, esso passa via via da una regola a quella successiva, sino a che esse non siano esaurite. Le regole che in base all'header del pacchetto stabiliscono d'applicare una politica di accept inviano il pacchetto direttamente all'inizio del chain successivo.

A questo punto il kernel decide che fare di un pacchetto arrivato alla fine chain in base alle politiche generali di default che vi sono state configurate. Normalmente le regole all'interno del chain stabiliscono quali pacchetti debbano essere rifiutati, mentre la politica di default a fine catena è di accept, così che il pacchetto non filtrato passi al vaglio del gruppo di regole successivo. Naturalmente è possibile anche adottare la logica inversa.

Il checksum serve a testare se il pacchetto è stato in un qualche modo corrotto. In caso affermativo viene bloccato.

Per semplificare, prima di ogni chain di regole viene effettuato un sanity check per evitare che dei pacchetti malformati possano confondere le rule. Qualora il sanity check riscontri qualcosa di anomalo, viene invocata una chiamata *syslog()*, per cui è possibile tenere traccia di eventuali problemi attraverso il syslogd.

L'input chain è il primo dei tre insiemi di regole che filtreranno il pacchetto; se l'esito non è drop o reject il pacchetto va avanti. A questo punto, se il filtro ha a che fare con la risposta ad un pacchetto mascherato, esso viene demascherato ed inviato direttamente all'output chain con un header riportante il destinatario reale.

Negli altri casi, in base al destination field del frame il routing code del kernel decide se mandarlo al trattamento locale (local process), oppure effettuare il forward del pacchetto ad un indirizzo remoto.

Nel primo caso un processo interno alla macchina riceve il pacchetto e lo manda all'output chain, che lo indirizza attraverso l'interfaccia di loopback, di modo che ricominci il ciclo con l'input chain settato sull'interfaccia lo.

Nel secondo caso il pacchetto non è stato generato da un processo locale, e può essere destinato ad un'altra entità sulla rete oppure al filtro stesso; il forward chain opera questa destinzione prima di inviarlo all'output chain.

Qualora il filtraggio sia effettuato su diversi frammenti, solo dopo di esso il kernel riassembla il pacchetto complessivo per il trattamento dei dati.

1.3 Esempi pratici di packet filter: il firewall

Redirezione: abbiamo una macchina con una sola scheda di rete. Su tale macchina non girano servizi importanti, ma vogliamo che eventuali interrogazioni a certe porte vengano inviate ad un'altra macchina. Questo processo si chiama redirezione o forwarding delle porte.

Firewall semplice: supponiamo che la nostra rete personale sia parte di Internet. I pacchetti possono passare dall'intranet all'extranet senza modifiche, tuttavia per motivi di sicurezza non si desidera che tutti i pacchetti possano filtrare da una rete all'altra. Inoltre talune macchine della nostra intranet non devono poter effettuare transazioni di dati con macchine all'esterno, e tuttavia devono aver la facoltà di rispondere.

Masquerading: la nostra intranet sta su una rete privata, disponiamo di un solo indirizzo internet per uscire, ciascuna delle nostre macchine deve quindi mascherarsi con tale indirizzo. Il firewall, cioè, non deve solo filtrare i pacchetti, ma modificarne l'header perché all'esterno appaiano tutti trasmessi dallo stesso indirizzo e le risposte vengano forwardate alla giusta destinazione. Questa operazione si chiama masquerading.

Limitato accesso dall'esterno alla intranet: indipendentemente dal fatto che sia configurato il masquerading o no, è possibile consentire un accesso limitato alla intranet tramite un redirettore di porte, o tramite un forward configurato sul firewall. In questo modo, se il firewall riceve dall'esterno una interrogazione diciamo sulla porta 80, destinata al web, esso la rigira sul web server della intranet, e poi riforwarda la risposta.

Transparent proxy: la rete privata non deve assolutamente comunicare coll'esterno. Sul firewall è installato un proxy server, ed un DNS. Le macchine della mia intranet non hanno configurato un default route, e per ogni connessione che cercano di effettuare all'esterno, questa viene forwardata al firewall sul proprio proxy, che si occupa della comunicazione reale e fornisce la risposta alla intranet. Questo è il solo caso in cui tutti i servizi fondamentali, mail, DNS, proxy, socket proxy, ecc. debbono essere posizionati sul firewall stesso.

1.4 Consigli pratici

Il packet filter opera sfruttando il socket buffer del kernel. A seconda del tipo di architettura ed a seconda della banda passante impiegata conviene configurare il buffer delle schede per evitare overflow, che comporterebbero una corruzione di parte dei pacchetti manipolati.

Su piattaforma Intel il default buffer per una scheda di rete è di 64K, che è l'ideale per una macchina che interagisce con una lan per operazioni standard. Un firewall sfrutta i buffer delle schede molto più pesantemente, quindi conviene accrescerlo, ma purtroppo non esiste una regola pratica per indicare di quanto debba essere aumentato. I valori cambiano a seconda del tipo di interfaccia e del tipo di rete. Per avere un'idea della scala dei valori si può dire che dopo varie prove condotte qualche anno addietro alla Scuola Normale Superiore di Pisa, per la loro rete abbiamo impostato un buffer di 256K per l'interfaccia rivolta alla lan a 10Mb, 512K per quella sull'anello a 100Mb, 512K per l'FDDI.

Per modificare il buffer si deve operare un echo del valore in byte rediretto nei file posti in **/proc/sys/net/core**

```
rmem_default    default setting of the socket receive buffer
wmem_default    default setting of the socket send buffer
rmem_max        maximum receive socket buffer size in bytes
wmem_max        maximum send socket buffer size in bytes
optmem_max      maximum ancillary buffer size allowed per socket
```

Tutte queste operazioni servono a migliorare la gestione dell'hardware da parte del sistema. Per migliorare la gestione dei pacchetti, invece, si deve configurare correttamente l'*MTU* e l'*MRU* delle singole interfacce, in base al supporto di rete ed alla finestra della banda passante, per ottimizzare il throughput sulle schede. Se la passante è buona non ci dovrebbero essere grossi problemi, invece con una banda ridotta è meglio tenere basso l'MRU (512) e settare l'MTU un po' più alto (1524), e configurare la **txqueuelen**, cioè la lunghezza della queue trasmissibile da una device fra il 4 ed il 10. Queste operazioni vanno compiute per mezzo del comando **ifconfig** al momento dell'inizializzazione dei supporti di rete.

Le seguenti impostazioni via procfs, invece, hanno lo scopo di migliorare un poco la sicurezza del firewall.

```
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
echo 1 > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
echo 1 > /proc/sys/net/ipv4/conf/all/secure_redirects
echo 1 > /proc/sys/net/ipv4/conf/all/send_redirects
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
echo 1 > /proc/sys/net/ipv4/conf/all/accept_source_route
echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
echo 0 > /proc/sys/net/ipv4/conf/all/log_martians
echo 1 > /proc/sys/net/ipv4/conf/all/rp_filter
```

IPCHAINS (kernel 2.2.x)

Nei kernel 2.2 per configurare il packet filter si usa il comando ipchains. Il manuale di questo comando, invocabile, come sempre, col comando **man**, è ben scritto ed estremamente facile da consultare. Per questo motivo qui vengono forniti esempi di implementazione che possono risultare utili sia come modello generale, sia per mostrare come si debbano combinare le varie opzioni.

Opzioni generali:

-N crea un nuovo chain
-X cancella un chain vuoto
-P muta la politica di un chain
-L elenca le regole in un chain
-F azzera le regole in un chain
-Z azzera il packet ed il byte counter in tutte le regole di un chain

Opzioni per manipolare le regole:

-A aggiungi in coda una nuova regola in un chain
-I inserisci una nuova regola in una posizione data
-R sostituisci una regola in una posizione data
-D cancella una regola in una posizione data, oppure la prima regola corrispondente alla descrizione data

Opzioni del masquerading:

-M riferisci le regole al masquerading
-S imposta in timeout per il masquerading

Opzioni per la manipolazione degli header dei pacchetti:

-s source address indirizzo/netmask
-d destination address indirizzo/netmask
-p protocollo tcp/udp/icmp/all
-i interfaccia ricevente
-j jump to policy *ACCEPT* / *DENY* / *REJECT* / *MASQ* / *REDIRECT* / *RETURN* / *FORWARD*

DENY blocca il pacchetto senza generare alcun ICMP di risposta, **REJECT** genera una risposta ICMP del tipo "destination unreachable". **MASQ** imposta il kernel affinché il pacchetto venga mascherato con l'indirizzo dell'interfaccia del filtro da cui viene poi rilasciato. **REDIRECT** opera una redirezione da una porta ad un'altra. **RETURN** invia il pacchetto direttamente al chain successivo. **FORWARD** opera il forwarding dei pacchetti.

-f applica la regola solo ai frammenti
-b modalità bidirezionale
-l log mode on
-o output max size
-v verbose nell'inizializzazione della regola

2.1 Operazioni su una singola regola

Ogni singola regola specifica un insieme di condizioni che il pacchetto deve onorare ed in base alle quali viene manipolato.

```
ipchains -N (nome chain)
```

Crea un nuovo chain. Solitamente non è necessario, a meno che non si voglia ottenere un filtraggio particolare, a costo di una maggiore lentezza nella gestione dei pacchetti. Il nome del chain non deve superare 8 caratteri.

```
ipchains -F (nome chain opzionale)
```

Svuota un chain, se il nome non è dato, vengono svuotati tutti i chains.

```
ipchains -X (nome chain)
```

Rimuove un chain vuoto cui non viene riferita alcuna regola.

```
ipchains -Z (nome chain)
```

Azzera il counter di un chain.

```
ipchains -L (nome chain opzionale)
```

Lista le regole contenute in un chain, se il nome non viene dato vengono listate tutte le regole.

```
ipchains -A (chain input/output/forward) -s (source address)
        -d (destination address) -p (protocol icmp/tcp/udp) port
                                                -j (ACCEPT/DENY)
```

In questo modo si aggiunge in coda ad un determinato chain una regola in relazione al filtraggio di una categoria di pacchetti. Solitamente si indica solo il source address oppure il destination address, solo in pochi casi (specialmente per i forward), entrambi.

```
ipchains -I (chain input/output/forward) (position)
                  -s (source address) -d (destination address)
         -p (protocol icmp/tcp/udp) port -j (ACCEPT/DENY/REJECT)
```

inserisci la regola nella posizione data (indicata numericamente).

```
ipchains -R (chain input/output/forward) -s (source address)
        -d (destination address) -p (protocol icmp/tcp/udp) port
                                          -j (ACCEPT/DENY/REJECT)
```

Muta la precedente politica di una regola (per esempio da ACCEPT a DENY).

```
ipchains -D ((chain input/output/forward) 1
```

cancella la prima regola in un chain

```
ipchains -D (chain input/output/forward) -s (source address)
         -d (destination address) -p (protocol icmp/tcp/udp) port
                                          -j (ACCEPT/DENY/REJECT)
```

Questo è un semplice mirror del comando con l'opzione -A e serve per cancellare una regola specifica.

Il source address ed il destination address possono essere indicati con nome da risolvere, con l'indirizzo numerico, con l'indirizzo numerico/netmask. Se il netmask non viene indicato il default è /32, ossia /255.255.255.255. Se non compare nella riga di comando nessun source o destination address il default diventa O/O che, naturalmente, viene interpretato come "da ogni indirizzo".

```
ipchains -A input -s 0/0 -j DENY
```

chiude ogni comunicazione.

La porta può essere indicata numericamente o col nome del servizio definito in /etc/services. Un arco di porte consecutive su indica con una notazione del tipo **6000:6010**, da 6000 a 6010 comprese.

"**!**" dopo l'indicazione del parametro, (-s ! localhost, ! www), rappresenta l'inversione, e viene interpretato come "eccetto".

ipchains non segue una logica ferrea. Si prenda il caso di un normale firewall, dotato di una schede di rete su internet ed una sulla intranet. A rigor di logica ci si aspetterebbe che nell'INPUT chain si filtrino i pacchetti indirizzati alle schede di rete, e nel forward chain quelli ruotati dal firewall per consetire la comunicazione fra le due reti che collega.

Non è così. Qualsiasi regola inserita nell'INPUT chain inpatta altresì anche i pacchetti instradati, per cui, se vi si filtra una porta, essa risulterà chiusa per il firewall stesso, ma anche per le macchine che vi si appoggiano per il routing.

Si noti che, affinché il routing venga effettuato, non si deve mai esplicitamente bloccare un icmp di type 3.

Se si desidera consentire una connessione TCP in una sola direzione, non si

deve dimenticare che, perché tale connessione possa aver luogo, è necessario uno scambio di pacchetti birezionale. Per consentire la sola ricezione dei SYN packet si deve usare l'opzione -y.

```
-p TCP -s (source address) -y
```

Tramite l'opzione -f è possibile specificare delle regole riferite esclusivamente ai frammenti successivi a quello iniziale di un pacchetto. Si badi, però, che tutti i pacchetti malformati sono trattati come frammenti, e solo i frammenti TCP in posizione 8 sono soggetti ad un drop, segnalato dal kernel con una chiamata kprintf, e reinviato dal klogd al syslogd. Per registrare il log su file il syslogd deve essere configurato per salvare "kern.info".

Un caso interessante è quello del masquerading. Come ovvio, il masquerading deve essere inserito all'interno del FORWARD chain. La linea di comando è:

```
ipchains -A FORWARD -s <intranet>/<netmask> -d 0.0.0.0/0 -j MASQ
```

Naturalmente le macchine mascherate possono comunicare con la rete esterna a loro piacimento, e riceverne i pacchetti di risposta. Eventuali macchine sulla rete esterna, invece non possono stabilire connessioni dirette ai client sottoposti a mascheratura. Infatti non saprebbero in che modo instradare i propri pacchetti. La necessità di far comunicare server esterni consentendogli di aprire una connessione con macchine poste dietro ad un firewall che le mascheri capita assai raramente. La difficoltà tecnica oggettiva viene superata solo con delle regole di REDIRECT.

In breve, si stabilisce che qualora un server dell'extranet cerchi di aprire una connessione con una porta specifica del firewall, quella connessione venga ridiretta su una determinata macchina dell'intranet.

Le regole di redirezione debbono essere inserite all'interno del chain di INPUT, e va abilitata l'opzione CONFIG_IP_TRANSPARENT_PROXY durante la fase di compilazione del kernel. Se si vuole che tutto funzioni a dovere, non bisogna scordarsi di specificare al firewall di trattare anche i SYN packet. In caso contrario il primo pacchetto verrebbe instradato correttamente, ma la risposta non saprebbe dove andare.

La linea di comando è:

```
ipchains -A INPUT -s <server su extranet> --destination-port
                    <porta> -j REDIRECT <server su intranet>
```

Masquerading e redirezione sono applicazioni di una tecnica di manipolazione dell'header dei pacchetti che prende il nome di Network Address Translation (NAT). Il kernel 2.2 non supporta il NAT in tutte le sue caratteristiche, per cui teoricamente l'amministratore di rete si trova a dover fare i conti con limitazioni funzionali piuttosto consistenti.

Tuttavia bisogna considerare che dal punto di vista pratico il NAT viene usato quasi esclusivamente per effettuare il masquerading, tanto che i due termini nel linguaggio tecnico italiano corrente sono erroneamente divenuti sinonimi.

2.2 Manipolazione del TOS

Solo uno dei quattro bit del TOS può essere settato. Per ovvie ragioni, su una rete con una banda passante ridotta conviene impostare il Minumum Delay per telnet ed ftp control, il Maximum Throughput per l' FTP-data, di modo da avere prestazioni accettabili quando si sia aperta più di una transizione di dati. Si deve usare a questo scopo l'opzione -t, secondo lo schema

Nome	Valore	Usi tipici
Ritardo minimo	0x01 0x10	ftp, telnet
Throughput massimo	0x01 0x08	ftp-data
Affidabilità massima	0x01 0x04	snmp
Costo minimo	0x01 0x02	nntp

Quindi si ha:

```
ipchains -A output -p tcp -d 0.0.0.0/0 telnet -t 0x01 0x10
ipchains -A output -p tcp -d 0.0.0.0/0 ftp -t 0x01 0x10
ipchains -A output -p tcp -s 0.0.0.0/0 ftp-data -t 0x01 0x08
```

Naturalmente questo riguarda i soli pacchetti in uscita, a meno che non si voglia usare un protocollo del tipo RSVP.

2.3 Logging

L'opzione -l attiva il logging, che consiste in una serie di messaggi dal kernel per mezzo della chiamata syslog(). Solitamente non conviene attivare il logging in quanto rallenta la manipolazione dei pacchetti, a meno che non si tratti di situazioni speciali o di categorie di pacchetti particolari, come ad esempio quelli provenienti da siti inseriti nelle liste di DENY o REJECT.

Il formato di output del logging è piuttosto complesso, ma fornisce molte infor-

mazioni riguardo al pacchetto loggato.

```
Packet log: input DENY eth0 PROTO=17 (source address):port
      (destination address):port  L=34 S=0x00 I=18 F=0x0000 T=254
                              #(posizione della regola nel chain)
```

input è il chain in cui viene effettuato il log.
DENY è il risultato del filtraggio del pacchetto.
PROTO è il protocollo cui appartiene il pacchetto. Viene indicato numericamente secondo le corrispondenze stabilite nel file /etc/protocols. I più comuni: 1 ICMP, 6 TCP, 17 UDP
(source address): port se si tratta di un pacchetto icmp, invece della porta è indicato il type
(destination address): port se si tratta di un pacchetto icmp, invece della porta è indicato il type
L numero di byte contenuti nel pacchetto
S bit del ToS del pacchetto, indicato in esadecimale
I IP id indicato numericamente
F bit del Fragment offset plus flags, indicato in esadecimale.
0x4 o **0x5** "Don't Fragment" indicano che non ci sono frammenti
0x2 o **0x3** "More Fragment" indicano che altri frammenti devono essere ricevuti. Le ultime due cifre rappresentano l'offset del pacchetto diviso 8
T Time to Live del pacchetto

2.4 Esempio di file di configurazione

Al reboot della macchina le regole vengono tutte azzerate. Una volta creata una configurazione ottimale, per salvare le regole si usa il comando:

```
ipchains-save > /etc/ipchains.rules
```

e si aggiunge in **rc.local**

```
if [ -f /etc/ipchains.rules ]; then
 /sbin/ipchains-restore < /etc/ipchains.rules
 echo 1 > /proc/sys/net/ipv4/ip_forward
 echo "."
fi
```

Una soluzione più flessibile consiste nel creare un file di configurazione che

verrà eseguito ad ogni boot.

Quello di seguito riportato è un esempio di configurazione complessa per un firewall posto fra una intranet privata a medio livello di sicurezza ed una CDN con una banda passante ridotta (64/128K). Questa configurazione, quindi, effettua il masquerading dei pacchetti e gestisce tutte le porte dei servizi principali.

```
# Local Interface
# This is the interface that is your link to the world

LOCALIF="eth1"

# Internal Interface
# This is the interface for your local network
# NOTE: INTERNALNET is a *network* address. All host bits should be 0

INTERNALNET="192.168.200.0/255.255.255.0"

# -- Variable definition -
#
# Set the location of ipchains.

IPCHAINS="/sbin/ipchains"

# You shouldn't need to change anything in the rest of this section

LOCALIP=`ifconfig $LOCALIF | grep inet
                                 | cut -d : -f 2 | cut -d \  -f 1`
LOCALMASK=`ifconfig $LOCALIF | grep Mask | cut -d : -f 4`
LOCALNET="$LOCALIP/$LOCALMASK"

echo "Internal: $INTERNALNET"
echo "External: $LOCALNET"

REMOTENET="0/0"

# - Flush everything, start from scratch -

echo -n "Flushing rulesets.."

# Incoming packets from the outside network
$IPCHAINS -F input
echo -n "."
```

```
# Outgoing packets from the internal network
$IPCHAINS -F output
echo -n ".”

# Forwarding/masquerading
$IPCHAINS -F forward
echo -n ".”

echo "Done!”

# - Allow all connections within the network -

echo -n "Internal..”

$IPCHAINS -A input -s $INTERNALNET -d $INTERNALNET -j ACCEPT
$IPCHAINS -A output -s $INTERNALNET -d $INTERNALNET -j ACCEPT
echo -n "..”

echo "Done!”

# - Allow loopback interface -

echo -n "Loopback..”

$IPCHAINS -A input -i lo -s 0/0 -d 0/0 -j ACCEPT
$IPCHAINS -A output -i lo -s 0/0 -d 0/0 -j ACCEPT
echo -n "..”

echo "Done!”

# - Masquerading -

echo -n "Masquerading..”

# don’t masquerade internal-internal traffic
$IPCHAINS -A forward -s $INTERNALNET -d $INTERNALNET -j ACCEPT
echo -n ".”

# don’t Masquerade external interface direct
$IPCHAINS -A forward -s $LOCALNET -d $REMOTENET -j ACCEPT
echo -n ".”

# masquerade all internal IP’s going outside
$IPCHAINS -A forward -s $INTERNALNET -d $REMOTENET -j MASQ
```

```
echo -n "."

# set Default rule on MASQ chain to Deny
$IPCHAINS -P forward DENY
echo -n "."

# - Allow all connections from the network to the outside -

$IPCHAINS -A input -s $INTERNALNET -d $REMOTENET -j ACCEPT
$IPCHAINS -A output -s $INTERNALNET -d $REMOTENET -j ACCEPT
echo -n ".."

echo "Done!"

# - Set telnet, www and FTP for minimum delay -
# This section manipulates the Type Of Service (TOS) bits of the
# packet. For this to work, you must have CONFIG_IP_ROUTE_TOS
# enabled in your kernel

echo -n "TOS flags.."

$IPCHAINS -A output -p tcp -d 0/0 www -t 0x01 0x10
$IPCHAINS -A output -p tcp -d 0/0 telnet -t 0x01 0x10
$IPCHAINS -A output -p tcp -d 0/0 ftp -t 0x01 0x10
echo -n "..."

# Set ftp-data for maximum throughput
#$IPCHAINS -A output -p tcp -d 0/0 ftp-data -t 0x01 0x08
#echo -n "."

echo "Done!"

# - Trusted Networks -
# Add in any rules to specifically allow connections from
# hosts/nets that would otherwise be blocked.

# echo -n "Trusted Networks.."

# $IPCHAINS -A input -s [trusted host/net]
                                  -d $LOCALNET <ports> -j ACCEPT
# echo -n "."

# echo "Done!"
```

```
# -- Banned Networks -
# Add in any rules to specifically block connections from
# hosts/nets that have been known to cause you problems. These
# packets are logged.

# echo -n "Banned Networks.."

# This one is generic
# $IPCHAINS -A input -l -s [banned host/net]
                                    -d $LOCALNET <ports> -j DENY
# echo -n "."

# This one blocks ICMP attacks
# $IPCHAINS -A input -l -b -i $LOCALIF -p icmp
                               -s [host/net] -d $LOCALNET -j DENY
# echo -n "."
# echo "Done!"

# - Specific port blocks on the external interface -
# This section blocks off ports/services to the outside that
# have vulnerabilities. This will not affect the ability to use
# these services within your network.
echo -n "Port Blocks.."

# NetBEUI/Samba
$IPCHAINS -A input -p tcp -s $REMOTENET -d $LOCALNET 139 -j DENY
$IPCHAINS -A input -p udp -s $REMOTENET -d $LOCALNET 139 -j DENY
echo -n "."

# Microsoft SQL
$IPCHAINS -A input -p tcp -s $REMOTENET -d $LOCALNET 1433 -j DENY
$IPCHAINS -A input -p udp -s $REMOTENET -d $LOCALNET 1433 -j DENY
echo -n "."

# Postgres SQL

$IPCHAINS -A input -p tcp -s $REMOTENET -d $LOCALNET 5432 -j DENY
$IPCHAINS -A input -p udp -s $REMOTENET -d $LOCALNET 5432 -j DENY
echo -n "."

# Network File System
$IPCHAINS -A input -p tcp -s $REMOTENET -d $LOCALNET 2049 -j DENY
$IPCHAINS -A input -p udp -s $REMOTENET -d $LOCALNET 2049 -j DENY
echo -n "."
```

```
# X Displays :0-:2-
$IPCHAINS -A input -p tcp -s $REMOTENET
                                    -d $LOCALNET 5999:6003 -j DENY
$IPCHAINS -A input -p udp -s $REMOTENET
                                    -d $LOCALNET 5999:6003 -j DENY
echo -n "."

# X Font Server :0-:2-
$IPCHAINS -A input -p tcp -s $REMOTENET -d $LOCALNET 7100 -j DENY
$IPCHAINS -A input -p udp -s $REMOTENET -d $LOCALNET 7100 -j DENY
echo -n "."

# Back Orifice (logged)
$IPCHAINS -A input -l -p tcp -s $REMOTENET
                                        -d $LOCALNET 31337 -j DENY
$IPCHAINS -A input -l -p udp -s $REMOTENET
                                        -d $LOCALNET 31337 -j DENY
echo -n "."

# NetBus (logged)
$IPCHAINS -A input -l -p tcp -s $REMOTENET
                                  -d $LOCALNET 12345:12346 -j DENY
$IPCHAINS -A input -l -p udp -s $REMOTENET
                                  -d $LOCALNET 12345:12346 -j DENY
echo -n "."
echo "Done!"

# -- High Unprivileged ports -
# These are opened up to allow sockets created by connections
# allowed by ipchains

echo -n "High Ports.."

$IPCHAINS -A input -p tcp -s $REMOTENET
                                 -d $LOCALNET 1010:65535 -j ACCEPT
$IPCHAINS -A input -p udp -s $REMOTENET
                                 -d $LOCALNET 1010:65535 -j ACCEPT
echo -n "."
echo "Done!"

# -- Basic Services -
echo -n "Services.."
```

```
# ftp-data (20) and ftp (21)
$IPCHAINS -A input -p tcp -s $REMOTENET -d $LOCALNET 20 -j ACCEPT
$IPCHAINS -A input -p tcp -s $REMOTENET -d $LOCALNET 21 -j ACCEPT
echo -n "..”

# ssh (22) (3000)
$IPCHAINS -A input -p tcp -s $REMOTENET -d $LOCALNET 22 -j ACCEPT
$IPCHAINS -A input -p tcp -s $LOCALNET -d $REMOTENET 22 -j ACCEPT
$IPCHAINS -A input -p tcp -s $REMOTENET -d $LOCALNET 3000 -j ACCEPT
$IPCHAINS -A input -p udp -s $REMOTENET -d $LOCALNET 3000 -j ACCEPT
$IPCHAINS -A input -p tcp -s $LOCALNET -d $REMOTENET 3000 -j ACCEPT
$IPCHAINS -A input -p udp -s $LOCALNET -d $REMOTENET 3000 -j ACCEPT
echo -n “.....”

# telnet (23)
# $IPCHAINS -A input -p tcp -s $REMOTENET -d $LOCALNET 23 -j ACCEPT
# echo -n “.”

# smtp (25)
$IPCHAINS -A input -p tcp -s $REMOTENET -d $LOCALNET 25 -j ACCEPT
echo -n “.”

# DNS (53)
$IPCHAINS -A input -p tcp -s $REMOTENET -d $LOCALNET 53 -j ACCEPT
$IPCHAINS -A input -p udp -s $REMOTENET -d $LOCALNET 53 -j ACCEPT
echo -n “..”

# DHCP on LAN side (67/68)
# $IPCHAINS -A input -i $INTERNALIF -p udp -s $REMOTENET
                              -d 255.255.255.255/24 67 -j ACCEPT
# $IPCHAINS -A output -i $INTERNALIF -p udp -s $REMOTENET
                              -d 255.255.255.255/24 68 -j ACCEPT
# echo -n “..”

# http (80)
$IPCHAINS -A input -p tcp -s $REMOTENET -d $LOCALNET 80 -j ACCEPT
echo -n “.”

# POP-3 (110)
$IPCHAINS -A input -p tcp -s $REMOTENET -d $LOCALNET 110 -j ACCEPT
$IPCHAINS -A input -p tcp -s $LOCALNET -d $REMOTENET 110 -j ACCEPT
echo -n “..”

# identd (113)
```

```
$IPCHAINS -A input -p tcp -s $REMOTENET -d $LOCALNET 113 -j ACCEPT
echo -n "."

# nntp (119)
$IPCHAINS -A input -p tcp -s $REMOTENET -d $LOCALNET 119 -j ACCEPT
echo -n "."

# https (443)
$IPCHAINS -A input -p tcp -s $REMOTENET -d $LOCALNET 443 -j ACCEPT
echo -n "."

# ICQ Services (it's a server service) (4000)
# $IPCHAINS -A input -p tcp -s $REMOTENET
                                      -d $LOCALNET 4000 -j ACCEPT
# echo -n "."
echo "Done!"

# -- ICMP -
echo -n "ICMP Rules.."

# Use this to deny ICMP attacks from specific addresses
# $IPCHAINS -A input -b -i $EXTERNALIF -p icmp
                                      -s <address> -d 0/0 -j DENY
# echo -n "."

# Allow incoming ICMP
$IPCHAINS -A input -p icmp -s $REMOTENET -d $LOCALNET -j ACCEPT
$IPCHAINS -A input -p icmp -s $REMOTENET -d $LOCALNET -j ACCEPT
echo -n ".."

# Allow outgoing ICMP
$IPCHAINS -A output -p icmp -s $LOCALNET -d $REMOTENET -j ACCEPT
$IPCHAINS -A output -p icmp -s $LOCALNET -d $REMOTENET -j ACCEPT
$IPCHAINS -A output -p icmp -s $INTERNALNET -d $REMOTENET -j ACCEPT
$IPCHAINS -A output -p icmp -s $INTERNALNET -d $REMOTENET -j ACCEPT
echo -n "...."
echo "Done!"

# -- set default policy -
$IPCHAINS -A input -j DENY
$IPCHAINS -A output -j ACCEPT

echo ""
echo "Finished Establishing Firewall."
```

NETFILTER (kernel 2.4.x 2.6.x)

Ipchains è una soluzione estremamente valida, ma, come si è visto, ha parecchie limitazioni. Poiché l'intera implementazione è a livello di kernel monolitico, qualsiasi modifica nel trattamento dei socket comporta una ricompilazione dell'intero kernel. Il codice non è modulare, né estensibile. Inoltre, per sua stessa natura, questa soluzione non permette delle politiche di filtraggio dinamiche.

Il sanity check che deve essere operato ad ogni passaggio da un chain all'altro rallenta notevolmente la manipolazione dei pacchetti.

Per ogni singolo pacchetto o frammento deve essere creato un socket

La redirezione non è il grado di gestire del tutto correttamente l'UDP; è possibile che un REDIRECT di un pacchetto UDP venga mandato su una porta shadowed.

Il forward ha informazioni inerenti solo all'interfaccia di uscita; per quella d'entrata l'amministratore deve avere una conoscenza esatta della topografia della rete. Un pacchetto forwardato deve passare comunque per tutti i chains senza possibilità di scavalcarli.

Non è possibile creare delle regole indipendentemente dall'interfaccia di rete che riceve i pacchetti.

Creare delle regole per una specifica interfaccia di rete è complesso.

Il masquerading viene effettuato durante il filtraggio. Questo significa che:

- all'input chain il SYN packet appare destinato al localhost.

- il forward chain non è in grado di manipolare pacchetti demasquerated
- l'output chain vede i pacchetti come se fossero provenienti dal localhost.

La manipolazione del TOS, la spedizione degli ICMP di type 03 in caso di REJECT, è affidata al filtro stesso, il che può disturbare il port forwarding.

Non c'è supporto per il NAT né per virtual servers.

Manca la possibilità di sfruttare appieno una feature fondamentale per un firewall moderno, il packet tracking. Grazie ad esso il kernel tiene traccia dei pacchetti che sono passati attraverso una macchina, in modo tale da poter capire come essi siano correlati fra loro all'interno delle varie connessioni.

Il processo di demascheratura, di cui abbiamo parlato in precedenza, è un esempio di packet tracking. Tale feature, però, offre possibilità interessanti anche al di fuori del NAT, in quanto, come vedremo, grazie ad essa si possono impostare dei filtri il base allo stato della connessione di cui fanno parte i pacchetti.

Per ovviare a questi inconvenienti i kernel 2.4 e 2.6 adottano una strategia mista. Il packet filter viene comunque implementato nel kernel, ma la parte di filtraggio avviene tramite dei moduli. Il nuovo sistema è stato concepito per essere dinamicamente estensibile grazie alla possibilità di operare dei download al volo dei moduli stessi che implementano le singole feature ed ad un sistema di shared library per il software gestionale. In tal modo è possibile mantenere l'efficienza di un filtering operato dal kernel in area non rimappabile, e tuttavia si acquista la dinamicità di una soluzione software.

Questo sistema si chiama netfilter oppure, dal nome del comando, iptables. Poiché l'albero delle directory contenente i moduli è stato notevolmente modificato, solo a partire dagli ultimi rilasci del kernel 2.4 è divenuto possibile il caricamento automatico tramite il kernel thread kmod, per abilitare i singoli moduli, coi kernel meno recenti occorre andare in

```
/lib/modules/<2.4.X>/kernel/net/ipv4/netfilter/
```

ed usare manualmente il comando modprobe.

Nel kernel è incluso anche un modulo di back-compatibility col sistema ad ipchains, caricando il quale si può ancora usare col vecchio tipo di packet filter a livello kernel.

3.1 La nuova filosofia implementativa

Nel nuovo tipo di netfilter le tre catene di filter sono destinate esclusivamente a contenere le regole che stabiliscono se un pacchetto debba essere accettato oppure rifiutato. Tutte le regole che comportano una modifica dell'header dei pacchetti debbono ora filtrare i pacchetti stessi prima o dopo che questi attraversino i tre chain "di filtraggio".

Per questo motivo sono stati creati tre ulteriori chain, uno di PREROUTING, prima dei tre chain di filter, uno di POSTROUTING dopo di essi e, alla fine del ciclo di filtraggio, un chain di OUTPUT.

Inoltre, dal momento che l'header dei pacchetti può essere modificato sostanzialmente con due scopi, operare una Network Address Translation oppure modificarne alcune proprietà, come ad esempio il TOS o aggiungere un marcatore (packet mangling), da un punto di vista logico si è deciso di tenere separate le due azioni.

I chain di PREROUTING, POSTROUTING ed OUTPUT sono di fatto duplicati a seconda se viene operato il NAT od il MANGLE dei pacchetti.

Per quanto riguarda il Network Address Translation è stata introdotta la distinzione fra nat vero e proprio, masquerade e forward. Ogni chain è dedicato ad un tipo particolare di regole:
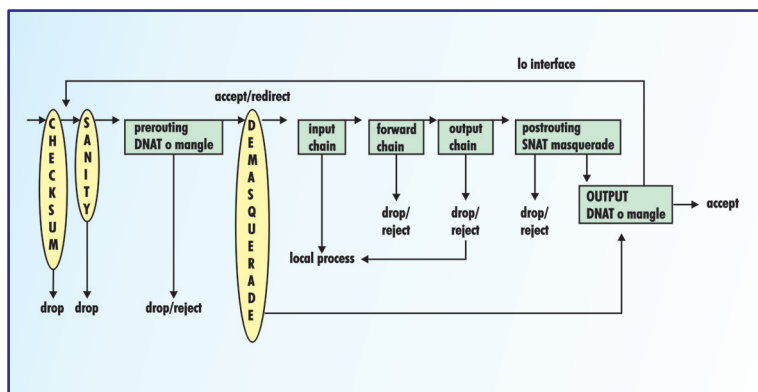
Figura 1: NAT layer

- Il chain di PREROUTING è destinato al Destination NAT, ossia la modifica del destination address dei pacchetti, al packet forward
- Il chain di POSTROUTING è destinato al Source NAT, ossia la modifica del destination address dei pacchetti ed al masquerade
- Il chain di OUTPUT è destinato al Destination NAT, ossia la modifica del destination address dei pacchetti, al packet forward

3.2 "iptables", il successore di "ipchains"

Per operare il packet filter avvalendosi anche del connection tracking il kernel deve caricare almeno questi moduli .

```
modprobe iptable_filter
modprobe ip_conntrack
modprobe ip_conntrack_ftp
modprobe ip_conntrack_tftp
modprobe ip_conntrack_amanda
modprobe ip_conntrack_irc
modprobe ip_tables
modprobe ip_queue
modprobe ipt_multiport
modprobe ipt_mac
modprobe ipt_unclean
modprobe ipt_REJECT
modprobe ipt_state
modprobe ipt_owner
modprobe ipt_limit
```

La gestione del filtering si effettua col comando iptables, la cui sintassi è identica a quella di ipchains. L'unica novità è la sostituzione, al posto del classico DENY, della politica DROP, che blocca il processo di filtraggio di un pacchetto all'interno di un chain, e cancella immediatamente il pacchetto stesso senza mandate nessun ICMP di type 03 come risposta.

Ci sono anche delle nuove politiche:

RETURN invia il pacchetto direttamente alla fine del chain. Se si tratta di un chain di sistema viene eseguita la politica del chain, se invece questo è stato creato dal sistemista, il pacchetto torna al chain precedente.
QUEUE invia il pacchetto all'userspace processing e, a meno che qualche applicazione non abbia necessità del pacchetto stesso, normalmente ne viene effettuato un DROP.

MIRROR inverte il contenuto dei campi source e destination nell'header di un pacchetto che poi viene ritrasmesso.

La sintassi della command line presenta, però, alcune restrizioni:

- Il nome dei built-in chains deve essere scritto sempre in maiuscolo, INPUT ed OUTPUT. Questi chains possono filtrare solo pacchetti generati e/o destinati localmente, mentre prima manipolavano tutti i pacchetti in entrata ed in uscita.
- L'opzione **-i** indica ora la sola interfaccia d'ingresso di un pacchetto, e funziona solo coi chains INPUT e FORWARD. Per i chains FORWARD ed OUTPUT al posto di **-i** si può anche usare **-o** con l'interfaccia di uscita. Tale opzione è l'unica usabile nella catena di OUTPUT.
- Le porte dei pacchetti TCP ed UDP devo ora essere indicate tramite **--source-port/--sport** e **--destination-post/--dport**, obbligatoriamente dopo l'indicazione **-p <protocol>**, che carica il modulo con l'estensione del protocollo nel kernel.
- Al posto di **-y** di deve utilizzare **--syn**, esclusivamente dopo **-p <protocol>**.
- È possibile azzerare il counter di un chain mentre questi sta filtrando un pacchetto.
- I nomi dei chains possono arrivare sino a 16 caratteri.
- MASQ e REDIRECT non sono più dei target, e sono demandati al NAT subsystem.

Anche il comando iptables è estensibile, le shared libraries contenenti le funzionalità aggiuntive si trovano in **/usr/lib/iptables**, per caricarle occorre l'opzione **-m**.

Quasi tutte le estensioni possono essere usate sia nelle tre catene di regole vere e proprie sia nel PREROUTING, POSTROUTING ed OUTPUT dei chain di NAT e MANGLE, alcune, però, per la loro natura hanno utilità solo se usate in un contesto specifico. In questo caso, per evitare inutili errori e confusioni, la command line rifiuta di usare l'estenzione laddove sia priva di senso.

Il comando

```
iptables -m <nome modulo> -h
```

lista le opzioni disponibili nella command line in relazione a ciascun modulo.

L'amministratore del firewall è comunque in grado di scrivere da sé le proprie estensioni, a seconda delle necessità contingenti utilizzando una apposita API fornita coi sorgenti di iptables, la libipt.

Esistono comunque parecchi moduli di default, ed il numero va aumentanto via via con l'evolversi del netfilter e della command line in user space.

```
-m mac --mac-source
```

Controlla il matching delle rule dei chain in base al source mac address. Per ovvi motivi non può essere usato nei chain di OUTPUT o POSTROUTING.. Naturalmente il mac address deve essere espresso nella forma **XX:XX:XX:XX:XX:XX**

Questa estensione è utilissima per attuare delle politiche specifiche sulle singole macchine in reti in cui gli indirizzi di rete sono assegnati tramite il dhcp.

```
-m limit
--limit <numero>
```

maximum average matching rate, indicato come numero e con una indicazione di tempo opzionale second minute hour day. Senza tale indicazione il default è hour.

```
--limit-burst <numero> massimo numero di pacchetti da controllare
```

Riduce il numero di match ad un certo limite per secondo, con un maximum burst prima che il limite venga comunque raggiunto. E' comodissimo col target LOG di modo da diminuire la produzione di log.

```
-m unclean
```

Esegue dei random sanity check sui pacchetti.

```
-m state --state NEW,ESTABLISHED,RELATED,INVALID
```

Se è stato caricato il modulo **ip_conntrack**, opera un match sullo stato dei pacchetti.

NEW	se il pacchetto crea una nuova connessione
ESTABLISHED	se il pacchetto appartiene ad una connessione preesistente
RELATED	se il pacchetto è correlato, ma non è parte di una connessione preesistene, ad esempio un ICMP che segnali un errore di comunicazione
INVALID	se non è stato possibile identificare il pacchetto

```
-m conntrack
--ctstate <stato>
--ctproto <protocollo>
--ctorigsrc [!] address[/mask]  (match sul source address originale)
--ctorigdst [!] address[/mask]  (match sul destination address originale)
--ctreplsrc [!] address[/mask]  (match sul reply source address)
--ctrepldst [!] address[/mask]  (match sul replu destination address)
--ctstatus [NONE|EXPECTED|SEEN_REPLY|ASSURED][,...]
--ctexpire time[:time]  (match la lifetime residua in secondi in base
                                 al valore o ai valori indicati)
```

Solo alcuni kernel supportano questo match che estende notevolmente le funzionalità del packet tracking, ben al di là del match sul tipo di connessione. Oltre ai quattro stati di cui abbiamo parlato per il modulo precedente, questo tipo di match opera su altri due stati.

SNAT se il source address originale è differente rispetto a quello presupposto dal pacchetti di replica

DNAT se il destination address originale è differente rispetto a quanto si potrebbe desumere dal pacchetti di replica

Con questi due stati è molto utile poter impostare gli indirizzi ip di riferimento (**--ctorigsrc --ctorigdst --treplsrc --trepldst**). **--ctstatus** si riferisce a come un pacchetto appaia all'interno del record delle connessioni.

```
-m helper
--helper <stringa>
```

effettua il match sui pacchetti correlati ad uno specifico conntrack-helper. La stringa potrebbe essere "ftp" per i pacchetti appartenenti ad una sessione ftp. Se si vuole far riferimento a porte differenti da quelle ufficiali per un determinato servizio, occorre indicarle in append alla stringa, usando "-" come separatore, ad esempio ftp-2121.

```
-m mark --mark (value)
```

Riconosce se è stato associato un determinato valore di mark ad una serie di pacchetti (si veda di seguito come settere il mark e perché è utile)

```
-m tos
--tos <tos>
```

Riconosce se nell'header IP sono stati impostati gli 8 bit che determinano un particolare Type of Service.

```
-m dscp
--dscp <valore 0/32>
--dscp-class <DiffServ Class>
```

Riconosce se sono stati impostati i 6 bit DSCP nell'header IP. Il DSCP dovrebbe nell'immediato futuro sostituirsi al TOS. Le classi DSCP possono essere BE, EF, AFxx or CSxx. Se si indica il valore del DSCP non se ne puo indicare al contempo la classe e viceversa.

```
-m multiport
--source-port [port[,port]]
--destination-port [port[,port]
--port [port[,port]]
```

Consente di indicare sino a 15 differenti porte tcp ed udp. Si ricordi di indicare anche che i pacchetti sono TCP od UDP con **-p tcp | udp**, in caso contrario la command line verrà rifiutata.

```
-m owner
--uid-owner userid
--gid-owner groupid
--pid-owner processid
--sid-owner sessionid
```

Permette di ricercare pacchetti creati da un processo con owner dalle caratteristiche (**id**, **gid**, **pid** e **session id** del processo creante), specificate con le opzioni sopra indicate; può essere usato esclusivamente nell'OUTPUT chain, e con alcuni pacchetti privi di owner quali, ad esempio, alcuni tipi di ICMP non è utilizzabile. Naturalmente è possibile controllare queste caratteristiche esclusivamente per i pacchetti generati localmente, per cui le regole con quest'estensione devono essere inserite nell'OUTPUT chain. Alcuni pacchetti, comunque, come gli ICMP-reply non hanno alcuna delle caratteristiche sopra indicate, per cui non possono essere riconosciuti da quest'estensione.

```
-m ah
--ahspi [!] spi[:spi]
```

Fa sì che la regola venga applicata agli SPI degli header AH dei pacchetti IPSEC

```
-m esp
--espspi [!] spi[:spi]
```

Fa sì che la regola venga applicata agli SPI degli header ESP dei pacchetti IPSEC

```
-m ttl
--ttl <valore>
```

Applica la regola solo ai pacchetti col Time to Live indicato.

```
-m length
--lenght <valore>
```

Applica la regola solo ai pacchetti di lunghezza indicata.

```
-m pkttype
--pkt-type [unicast|broadcast|multicast]
```

Applica la regola solo ai pacchetti con link layer del tipo indicato (unicast, broadcast, multicast)

```
-m physdev
--physdev-in <nome interfaccia>
--physdev-out <nome interfaccia>
```

Questo modulo è parte dell'infrastruttura che abilita i transparent bridge con i kernel 2.6. Infatti esso realizza il match su una device enslaved ad una bridge device. La device di input può venire indicata solo nelle catene di INPUT, FORWARD e PREROUTING, quella di output nei chain FORWARD, OUTPUT e POSTROUTING, ad eccezione di nat e mangle, solo nella chain di OUTPUT. Se il nome dell'interfaccia termina con un "+", allora la regola fa riferimento a tutte le schede che il cui nome inizi con la stringa indicata, ad esempio eth+ indica tutte le schede ethernet.

```
-m ecn
--ecn-tcp-cwr
--ecn-tcp-ece
-ecn-ip-ect [0..3]
```

Attiva la regola solo in relazione ai pacchetti che abbiano i bit di Explicit Congestion Notification configurati. **--ecn-tcp-cwr** opera il match sul bit CWR dell'header TCP. **--ecn-tcp-ece** opera il match sul bit ECE dell'header TCP. Infine **--ecn-ip-ect [0..3]** effettua il match sul codepoint nell'header IPv4, il valore può variare da un minimo di 0 ad un massimo di 3.

```
-m rpc
--rpcs list,of,procedures
--strict
```

Attiva la regola solo in relazione ai pacchetti inerenti programmi RPC. col parametro **-rpcs <list,of,procedures>** si può indicare il program number, sia in forma numerica, es. 100003, sia col nome, es. nfs.

La tabella con le corrispondenze fra numero identificativo e nome dei programmi RPC è contenuta nel file **/etc/rpc**. Il parametro **--strict** causa il drop dei pacchetti che non contengano una richiesta "get" al portmapper.

In base ai differenti match che si ottengono è possibile stabilire politiche separate di filtraggio per ciascuna singola porta.

Per tutte le opzioni aggiuntive delle estensioni sopra indicate è possibile usare "! " col significato di "tutti i pacchetti eccetto".

Anche la possibilità di discernere se un pacchetto appartenga al protocollo TCP UDP o ICMP è implementata per mezzo dei moduli. In questo caso, però, essi possono essere caricati oltre che con l'opzione **-m** anche con il tradizionale parametro **-p**. Il comando

```
iptables -p (protocol) -h
```

lista i flag o i type di un protocollo.

Anche in questo caso ogni estensione implementa delle opzioni particolari:

tcp: controlla i pacchetti TCP. Le opzioni sono:

--source-port <port[: port]>: indica la source port, abbreviato in **--sport**

--destination-port <port[: port]>: indica la destination port, abbreviato in **--dport**

--tcp-flags <flags[,flag]>: fa sì che la regola sia applicata solo ai pacchetti con settati nell'header i flag specificati. I flag sono: **SYN ACK FIN RST URG PSH ALL NONE**. L'opzione **--syn** è equivalente a **--tcp-flags SYN,RST,ACK SYN**

--tcp-option <numero>: fa sì che la regola sia applicata solo ai pacchetti con settato nell'header la TCP option indicata

udp: controlla i pacchetti UDP. Le opzioni sono:

--source-port <port[: port]>: indica la source port, abbreviato in **--sport**

--destination-port <port[: port]>: indica la destination port, abbreviato in **--dport**

icmp: controlla i pacchetti ICMP: Le opzioni sono:

--icmp-type <type>: applica la regola solo ai pacchetti ICMP del type specificato

```
iptables -p <tcp|udp|icmp> -h
```

Lista i flag o i type di un protocollo.

Anche in questo caso per tutte le opzioni aggiuntive delle estensioni sopra indicate è possibile usare "! " col significato di "tutti i pacchetti eccetto".

3.3 I tre chain di filtraggio visti nei dettagli; principi generali per la configurazione basilare di un firewall

La metodologia di funzionamento dei tre filter chain è stata notevolmente modificata. Se si vuol essere rigorosi, si può affermare senza ombra di dubbio che è stata resa molto più logica *(figura 2)*.

Prima di passare attraverso le catene di regole il kernel stabilisce se il pacchetto è destinato localmente oppure deve essere sottoposto a routing. Di conseguenza, se un pacchetto è destinato al firewall stesso viene dirottato all'INPUT chain, e se le regole lo autorizzano, arriva al POSTROUTING per poi venire processato localmente.

I pacchetti generati dal firewall, sia in risposta a quelli provenienti dall'esterno,
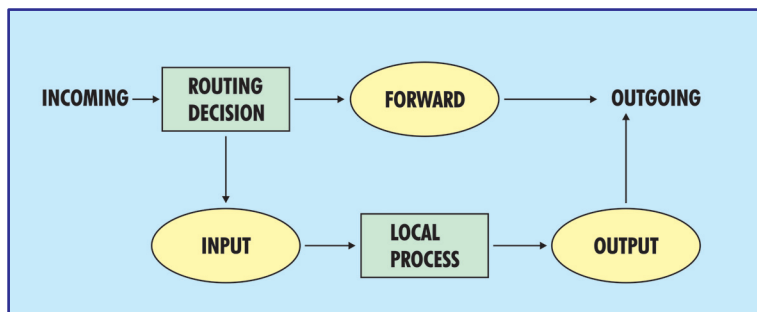
Figura 2: Internal chain layer

sia emessi autonomamente, prima di uscire dall'interfaccia di rete, vengono fatti passare per l'OUTPUT chain e da lì vanno al POSTROUTING.

Qualora il pacchetto non sia destinato localmente, bensì al routing, ossia debba passare attraverso il firewall per andare da una rete ad un'altra, viene mandato direttamente la FORWARD chain. Qui il pacchetto viene filtrato e, se le regole lo consentono, passa al POSTROUTING.

In questo modo ogni chain ha una funzione specifica:

INPUT: filtra i pacchetti destinati all'interfaccia locale
OUTPUT: filtra i pacchetti in uscita dall'interfaccia locale
FORWARD: filtra i pacchetti destinati al processo di routing

Se una regola stabilisce l'ACCEPT di un pacchetto, questi esce dal chain per andare immediatamente al POSTROUTING. Per questo motivo, qualora si voglia restare strettamente aderenti alla logica del netfilter, la policy di default di ogni catena del filter chain dovrebbe essere DROP, mentre è all'interno che ogni singolo servizio dovrebbe venir abilitato. Naturalmente è possibile anche il contrario, tuttavia nello spiegare il funzionamento dettagliato di ogni singolo chain ci atterremo al caso piú fedele alle intenzioni degli implementatori del Netfilter.

Ogni chain ha un suo comportamento peculiare. La catena di INPUT si divide in tre sottocatene. Quando arriva un pacchetto per prima cosa viene controllato il protocollo d'appartenenza, e tramite il controllo dell'header si stabilisce se è stato emesso dalla rete locale su cui si affaccia l'interfaccia di rete dalla quale è arrivato o direttamente dal localhost; inoltre si analizzano il source address e destination address, source port e destination port oppure icmp type, TTL, mark e tutti gli altri dati desumibili.

Successivamente il kernel deve capire se il pacchetto appartiene ad una connessione già stabilita, per cui viene marcato come appartenente alla categoria ESTABLISHED, oppure richiede l'inizio di una connessione in dipendenza da un'altra preesistente, ossia appartiene alla categoria RELATED.

In dettaglio: la categoria di ESTABLISHED si applica a tutti i flussi dati continui, siano essi TCP, come un normale telnet, UDP, quali le query DNS, od ICMP, ad esempio traceroute o ping. Il kernel intende il flusso dei pacchetti come bidirezionale, anche perché non ha modo di distinguere solo in base ad uno stream in una unica direzione quale delle due macchine si sia connessa e quale abbia

risposto.

RELATED è riferibile a parecchi tipi di connessione. L'esempio classico è il funzionamento del servizio FTP. La connessione primaria è sulla porta TCP 21, ma il flusso dati avviene con una RELATED connection sulla porta TCP 20. Ovviamente viene considerato RELATED anche il traffico ICMP generato dalle connessioni TCP.

Ci sono altre due categorie. I pacchetti che richiedono la creazione di una nuova connessione sono marchiati come NEW, e quelli non associati ad alcuna connessione vengono considerati INVALID.

I pacchetti ascrivibili alla categoria NEW sono, essenzialmente, di tipo TCP e si distinguono poiché hanno nell'header il bit di SYN settato e quello di ACK nullo.
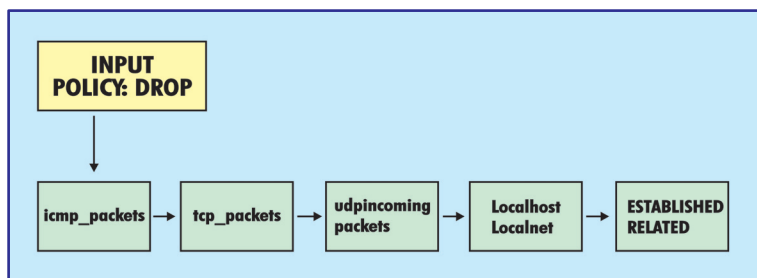
Figura 3: INPUT chain layer
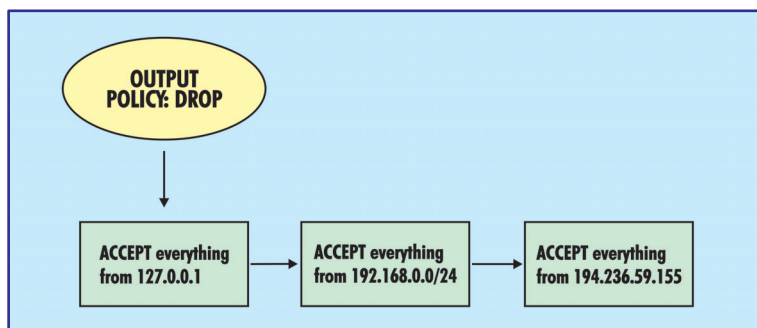
Figura 4: OUTPUT chain layer
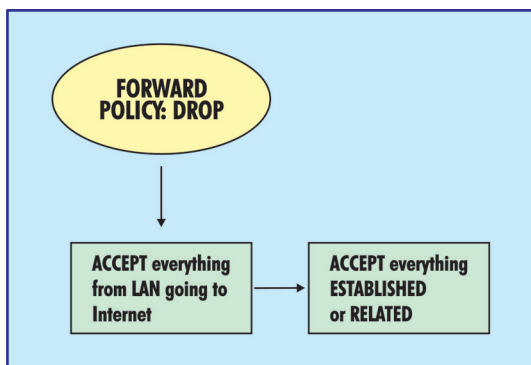
Figura 5: FORWARD chain layer

INVALID viene applicato ai pacchetti che hanno nell'header un set di opzioni invalido. XMAS-tree scan si basa appunto su questo tipo di pacchetti, e non è contrastabile in nessuna altra maniera, tanto che alla maggioranza degli altri tipi di firewall esistenti risulta a tutti gli effetti invisibile.

Solo a questo punto il pacchetto passa effettivamente per le regole impostate nel chain. Se una regola stabilisce l'ACCEPT il pacchetto esce dal chain per essere processato localmente. In caso di DROP il pacchetto viene droppato. Qualora nessuna regola filtri il pacchetto esso verrà manipolato a seconda della politica di default configurata al termine della catena *(figura 3)*.

L'OUTPUT chain filtra i pacchetti in uscita, generati dal localhost. Internamente segue una meccanica identica a quella dell'INPUT chain. Normalmente vi si inseriscono poche regole, che autorizzano l'uscita di pacchetti provenienti dall'indirizzo di loopback 127.0.0.1, e dai singoli indirizzi delle interfacce di rete *(figura 4)*.

La politica generale del chain è solitamente DROP. Per evitare che alcuni server su Internet si rifiutino di trattare correttamente i pacchetti frammentati, in questo chain si usa inserire una regola per cui l'MSS dei pacchetti, se troppo grande, venga ridotto alle dimensioni dell'MTU.

Nell'OUTPUT chain non è possibile usare la politica MIRROR.

Se un pacchetto è destinato ad essere ruotato da una rete ad un'altra il chain in cui viene filtrato è il FORWARD. In funzionamento è identico a quello descritto per le due catene precedenti. In un firewall, di solito, vengono accettati tutti i pac-

chetti provenienti dalla intranet indirizzati verso Internet. Per i pacchetti originari dall'esterno e diretti alla nostra rete locale, la miglior politica di protezione consiste nell'accettare solo quelli apparteneti alle classi ESTABLISHED e RELATED. Naturalmente è opportuno che tutti i pacchetti malformati siano soggetti a DROP. Buona norma, come è stato spiegato in precedenza, è anche assicurarsi che l'MSS dei pacchetti instradati abbia al massimo una grandezza corrispondente a quella dell'MTU *(figura 5)*.

3.4 Il connection tracking e la sua utilità

Il packet tracking, come si può facilmente osservare, viene applicato a tutti i protocolli di rete, nei limiti delle caratteristiche di ciascuno di essi, e, grazie alla sua architettura granulare, è un trattamento estremamente efficace.

Esso, infatti, consente di effettuare dei controlli anche sulla funzione dei pacchetti, oltre che sulla loro natura. I pacchetti tracciati vengono listati nel file virtuale **/proc/net/ip_contrack** e, quindi, in realtà la procedura può richiedere anche un quantitativo di memoria considerevole. Difficilmente una macchina con solo 16 mega di RAM è in grado di operare un packet tracking efficiente.

Al momento del boot, comunque, il kernel cerca di stabilire in base alla RAM totale disponibile un valore di default non troppo basso e neppure esageratamente alto. Questo valore, espresso in byte, è visualizzabile operando un cat del file **/proc/sys/net/ipv4/ip_conntrack_max**

Normalmente per 128 Mega di RAM vengono riservati 8 Kbyte. Tale valore non deve sembrare basso, è semplicemente la dimensione del buffer che contiene l'output testo degli ultimi pacchetti tracciati. Per incrementare o diminuire la quantità di memoria riservata al packet tracking è sufficiente agire su tale file con una semplice echo della misura in byte di RAM che si desidera utilizzare rediretto nel file.

```
echo 65536 > /proc/sys/net/ipv4/ip_conntrack_max
```

Il contenuto di questo buffer è accessibile via procfs, cosí da poter esse visualizzato tramite il comando:

```
cat /proc/net/ip_conntrack
```

L'output sarà simile a:

```
tcp    6 71 SYN_SENT src=213.45.1.167 dst=80.182.26.118
```

```
sport=34010 dport=4662 [UNREPLIED] src=80.182.26.118
                                                          dst=213.45.1.167
sport=4662 dport=34010 use=1
udp    17 8 src=217.227.53.108 dst=213.45.1.167 sport=9406
dport=10361 src=213.45.1.167 dst=217.227.53.108 sport=10361
dport=9406 [ASSURED] use=1
tcp    6 431999 ESTABLISHED src=127.0.0.1 dst=127.0.0.1
sport=32985 dport=4001 src=127.0.0.1 dst=127.0.0.1 sport=4001
dport=32985 [ASSURED] use=1
udp    17 28 src=213.45.1.167 dst=1.0.0.0 sport=10361
dport=8242 [UNREPLIED] src=1.0.0.0 dst=213.45.1.167 sport=8242
dport=10361 use=1
tcp    6 6 TIME_WAIT src=213.45.1.167 dst=194.177.206.124
sport=33914 dport=4662 src=194.177.206.124 dst=213.45.1.167
sport=4662 dport=33914 [ASSURED] use=1
udp    17 124 src=62.82.229.89 dst=213.45.1.167 sport=8421
dport=10361 src=213.45.1.167 dst=62.82.229.89 sport=10361
dport=8421 [ASSURED] use=1
udp    17 5 src=213.45.1.167 dst=61.62.86.61 sport=4666
dport=4246 [UNREPLIED] src=61.62.86.61 dst=213.45.1.167
sport=4246 dport=4666 use=1
```

Il formato è piuttosto semplice. In primo luogo viene indicato il protocollo, tcp udp o icmp, poi il suo valore espresso come numero decimale. Segue l'indicazione di quanto tempo, espresso in secondi, l'entry verrà ancora mantenuta in memoria dal kernel. Dopo la durata viene indicata, per i pacchetti TCP, la funzione del pacchetto all'interno del data flow cui appartiene.

Sono possibili i seguenti valori,

```
NONE 30 minuti
ESTABLISHED 5 giorni
ASSURED -
SYN_SENT 2 minuti
SYN_RECV 60 secondi
FIN_WAIT 2 minuti
TIME_WAIT 2 minuti
CLOSE 10 secondi
CLOSE_WAIT 12 ore
LAST_ACK 30 secondi
LISTEN> 2 minuti
```

la durata dei timeout esposta nella tabella corrisponde al default impostato nel kernel, ma è possibile modificare tali impostazioni agendo sul procfs negli opportuni file siti in **/proc/sys/net/ipv4**.

Fra parentesi quadre viene indicato se il kernel è ancora in attesa d'un pacchetto di reply a quello riportato (UNREPLIED), oppure se esso è già stato osservato (ASSURED).

Il base alla memoria dei pacchetti già ricevuti il kernel è in grado di stabilire quale sia la funzione di quelli nuovi in arrivo in relazione ai data flow in corso.

3.4.1 Connessioni TCP

Il connection tracking funziona più o meno allo stesso modo TCP UDP ed ICMP. Le connessioni TCP sono stateful. Occorre, però, tenere presente che per il kernel, nel caso del TCP, è sufficiente vedere il solo pacchetto con la richiesta di SYN per ritenere una connessione come NEW, mentre al passaggio della sequenza SYN/ACK essa acquisisce lo stato di ESTABLISHED.

In questo modo è possibile impostare delle regole estremamente granulari. Il firewall può quindi consentire che dalla intranet partano delle connessioni NEW ed ESTABLISHED, ed al contempo accettare dalla extranet solo i pacchetti di risposta a quelle ESTABLISHED. Lo stato di ESTABLISHED come quello di UNREPLIED, in verità, vengono solo presunti dal kernel. In ogni caso la creazione d'una connessione TCP comporta un handshake a 3 vie.

SYN SENT si riferisce ad una connessione TCP di cui è arrivato solo il primo pacchetto SYN in una sola direzione.
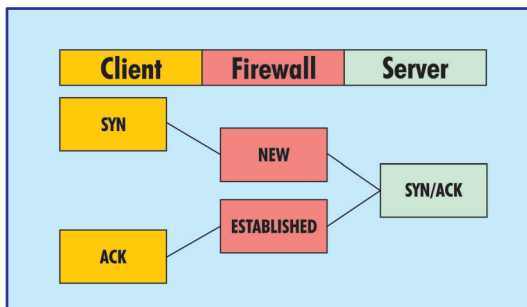
Figura 6: 3-way handshake di una connessione TCP
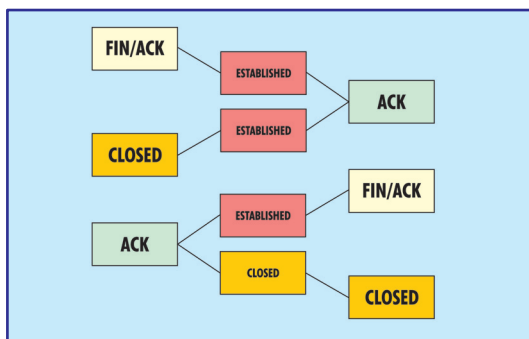
Figura 7: Chiusura di una connessione TCP

SYN RECV indica che la richiesta di connessione è stata onorata tramite il pacchetto SYN/ACK, che ha attraversato il firewall senza venire filtrato da alcuna regola.

Solo quando è stato ricevuto anche il pacchetto di final ACK la procedura può venir considerata come effettivamente terminata e la connessione acquisisce lo stato di ESTABLISHED. Dopo qualche altro pacchetto il kernel passa a considerare la connessione come ASSURED *(fig. 6).*

La chiusura di una connessione TCP si ottiene in condizioni normali tramite un pacchetto di FIN cui l'altro host risponde con un ACK. Solo quando anche questo ultimo ha attraversato il firewall la connessione viene considerata CLOSED, ed entra nello stato di TIME WAIT, la cui durata è al solito di due minuti *(figura 7)*. In questo modo tutti i pacchetti che arrivano per qualche ragione dopo la chiusura, ma si inseriscono all'interno di una connessione, vengono ancora trattati il base al rule-set impostato nel kernel. Non si tratta di un caso infrequente, anzi è normale quando si ha a che fare con un router congestionato.

Le la connessione viene chiusa brutalmente con un pacchetto di reset RST, lo stato diviene subito CLOSED, e dopo 10 secondi la connessione verrà del tutto tranciata.

3.4.2 Connessioni UDP

Le connessioni UDP sono per loro natura stateless. Di conseguenza non vi è alcun handshake per stabilire una connessione, né una vera e propria chiusura.
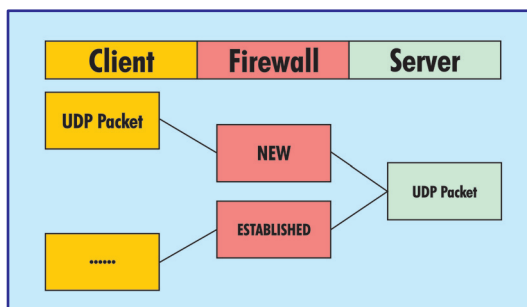
Figura 8: creazione di una connessione UDP
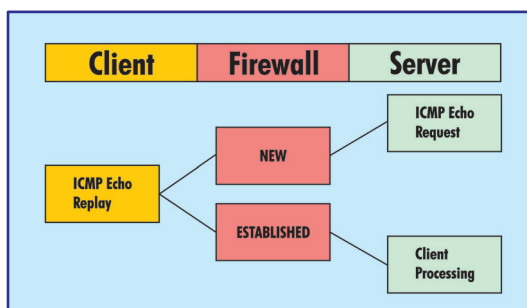
Figura 9: Non esistono delle vere e proprie connessioni ICMP, tuttavia il kernel ne tiene traccia con un meccanismo simile a quello usato per l'UDP

Inoltre l'ordine di ricezione dei datagrammi UDP solitamente non rispecchia quello di invio. Al passaggio del primo pacchetto UDP attraverso un flusso di dati, il kernel considera la connessione come UNREPLIED.

Il timeout per codesto stato è di 30 secondi.

Appena attraverso il firewall viene osservato il pacchetto di reply la connessione acquisisce lo stato di ESTABLISHED. Tuttavia, solo dopo il passaggio di un certo numero di pacchetti facenti capo ad una medesima connessione, il kernel passa a considerarla come ASSURED.

Se all'interno di una connessione non si verifica alcuno scambio di pacchetti per più di 180 secondi, essa viene considerata come terminata. 180 secondi

rappresenta un valore di timeout piuttosto basso, ma è sufficiente nella maggior parte dei casi. Talora il sistemista si trova comunque a doverlo aumentare.

3.4.3 Connessioni ICMP

L'ICMP è un protocollo che in sé e per sé non opera all'interno d'alcun tipo di connessione. Esso si basa esclusivamente sull'emissione da parte di un host di pacchetti con un type contenente una specifica request, cui l'host di destinazione fa seguito con altrettanti pacchetti con un type di reply alla funzione richiesta. Classico è l'esempio del ping, ad un icmp di *echo request (ping)*, viene inviato in risposta un icmp di *echo reply (pong)*.

Al passaggio del primo icmp di request, il kernel tiene traccia del pacchetto come se si trattasse di una connessione di stato NEW e considera il pacchetto stesso come UNREPLIED. Il timeout di default per questa condizione è di 30 secondi. La connessione diviene ESTABLISHED appena il firewall osserva il pacchetto di reply.

Nel caso dei pacchetti ICMP con type utilizzato per indicare ad un host lo stato di una determinata conessione TCP od UDP, il kernel considera il loro flusso come appartenente ad una connessione RELATED a quella TCP od UDP cui si riferiscono.

3.5 Poche regole per un firewall efficiente

Supponendo di utilizzare i moduli di connection tracking, una configurazione minimale dei chains di filter di un firewall diverrà simile a:

```
iptables -p INPUT DROP
iptables -p FORWARD DROP
iptables -p OUTPUT DROP

iptables -A INPUT -i <intranet net card> -p tcp --source-address
             <management station> --destination-port 22 -j ACCEPT

iptables -A INPUT -i <extranet net card> -m unclean -j DROP

iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

iptables -A INPUT -p icmp --icmp-type 8
                             -m length --length 128:65535 -j DROP

iptables -A FORWARD -i <extranet net card>
                   -m state --state RELATED,ESTABLISHED -j ACCEPT
```

```
iptables -A FORWARD -i <intranet net card>
               -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT

iptables -A FORWARD -p icmp --icmp-type 8
                             -m length --length 128:65535 -j DROP

iptables -A FORWARD -m pkttype --pkt-type multicast -j DROP

iptables -A FORWARD -m state --state INVALID
                                 -i <intranet net card> -j REJECT

iptables -A OUTPUT
               -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
```

A meno che non sia espressamente ordinato da una regola, ogni pacchetto che debba passare attraverso il firewall oppure giunga o parta da esso viene per default droppato. Fra gli altri vantaggi di questo approccio, in questo modo non è necessario impostare alcuna regola per il DROP delle connessioni INVALID.

In linea di principio, si consideri che un firewall non è una macchina destinata all'erogazione di servizi specifici, ad eccezione talvolta del DNS.

Di conseguenza ad eccezione dell'ssh, su porta 22 TCP, per la connessione dell'amministratore di sistema, e delle porte 53 su TCP ed UDP, ogni altro pacchetto che giunga all'INPUT chain dovrebbe essere sottoposto a DROP, a meno che non sia un syn packet o non appartenga alla categoria ESTABLISHED o RELATED.

Inoltre, il controllo sull'eventuale malformazione dei pacchetti protegge da alcune maliziose tecniche di hacking.

Si noti che nell'INPUT chain vengono impostate delle regole atte a controllare la length dei pacchetti ICMP di type echo request, utile contro alcuni tipi di DOS. Quest'ultima regola viene riproposta nel chain di FORWARD per impedire che simili attacchi possono passare attraverso il firewall ad opera di qualche utente malizioso della intranet o di qualche script kiddie dall'esterno.

Sempre nella catena di FORWARD vengono filtrati tutti i pacchetti di tipo multicast.

Se dalla intranet provengono pacchetti di categoria INVALID, invece del normale DROP viene restituito un icmp di "host not found".

Funzioni fondamentali quali il masquerade ed il NAT in generale, come si è detto, non sono contenute nei filter chain, bensì all'interno delle catene di nat e mangle.

3.6 NAT ed "iptables -t nat"

Il NAT *(Network Address Translation)*, è il processo tramite il quale un link di rete altera l'header dei pacchetti che gli passano attraverso tenendo traccia delle manipolazioni effettuate, ed operando il reverse natting sui pacchetti di risposta.

Il funzionamento della traslazione a livello kernel è piuttosto complesso ed è meglio spiegarlo dopo aver visto come operi la traslazione e come si possano impostarne le politiche.

Il NAT non è utile solo per fare masquerading e redirezionamento, esso permette anche di smistare i pacchetti in arrivo ad server multipli utilizzando un solo indirizzo IP .

Per utilizzare il NAT è necessario caricare gli appositi moduli.

```
modprobe iptable_nat
modprobe ip_nat_ftp
modprobe ip_nat_tftp
modprobe ip_nat_irc
modprobe ip_nat_snmp_basic
modprobe ip_nat_amanda
modprobe ipt_MASQUERADE
```

Il comando per configurare il NAT è **iptables -t nat**, ed ha delle opzioni molto simili ad iptables.

-I inserisci un nuovo NAT
-D cancella un NAT esistente
-p protocollo TCP/UDP
-s source address nell'header del pacchetto entrante, indirizzo/netmask è possibile indicare un raggio di indirizzi, separandoli con "-"
-d destination address nell'header del pacchetto entrante, indirizzo/netmask è possibile indicare un raggio di indirizzi, separandoli con "-"
--source-port source port nell'header del pacchetto entrante è possibile indicare un raggio di porte, separandole con "-"
--destination-port destination port nell'header del pacchetto entrante è possibile indicare un raggio di porte, separandole con "-"
-i incoming network interface
-o outcoming network interface

--to-destination nuovo destination address: porta da inserire nell'header del pacchetto.
--to-source nuovo source address: porta da cambiare nel pacchetto.
--to-ports destination port del reindirizzamento per la politica REDIRECT è possibile indicare un raggio di porte, separandole con "-"
-m null/masquerade/redirect/static tipo di mappaggio su un raggio di indirizzi

Le politiche possibili per la gestione dei pacchetti sono

DNAT: muta il destination address e la relativa porta nell'header di un pacchetto
SNAT: muta il source address e la relativa porta nell'header di un pacchetto
REDIRECT: consente di mutare la porta di destinazione di un pacchetto

Come si è già detto, il Destination NAT (DNAT) deve essere effettuato nei chain di PREROUTING e di OUTPUT, mentre il Source NAT (SNAT) in quello di POSTROUTING. Ecco alcuni esempi

```
iptables -t nat -A PREROUTING -p (protocol tcp/udp)
                        -s (source address) --source-port <port>
               -d (destination address) --destination-port <port>
 -j DNAT --to-destination (new header destination address):(port)
```

Muta il destination address e la porta di destinazione nell'header dei pacchetti che soddisfano la regola. Solo dopo tale modifica i pacchetti passano attraverso la catena che chain di filtro.

```
iptables -t nat -A POSTROUTING -p (protocol tcp/udp)
                        -s (source address) --source-port <port>
               -d (destination address) --destination-port <port>
           -j DNAT --to-source (new header source address):(port)
```

Dopo che i pacchetti hanno attraversato i chain di filtro, il source address dell'header di quelli che soddisfano la regola viene mutato; di conseguenza i pacchetti di risposta vengono inviati al nuovo indirizzo e non a quello di provenienza reale.
Per attivare il masquerading:

```
iptables -t nat -I POSTROUTING -s (intranet/netmask)
   -o (interfaccia sulla extranet) --source-address -j MASQUERADE
```

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Indicare espressamente l'interfaccia di output è molto utile per evitare le fastidiose metodologie di intrusione basate sullo spoofing.

Per effettuare una redirezione delle porte di destinazione di un pacchetto:

```
iptables -t nat -A PREROUTING -p (protocol tcp|udp)
                          -d (destination address in the header)
                 --destination-port (destination port in the header)
               -j REDIRECT -to-ports --to-port (port to redirect to)
```

In questo modo viene rediretto tutto il traffico per un dato host sulla porta indicata.

```
iptables -t nat -A PREROUTING -p (protocol tcp|udp)
  -s (source address in the header) -d (destination address in the header)
       --destination-port (destination port in the header)
   -j REDIRECT --to-ports (destination address to redirect to)
```

Permette di selezionare le redirezioni anche in base agli host da cui provengono i pacchetti.

3.7 Funzionamento del NAT

Come si può intuire da quanto è stato esposto, il NAT è basato sulla mappatura degli header dei pacchetti, nel senso che un header viene analizzato e le modifiche vengono mappate al suo interno in base, prima di tutto, agli eventi precedentemente verificatisi nella manipolazione dei pacchetti già filtrati, e solo successivamente in virtù delle rule.

Quando un pacchetto arriva, il kernel osserva tutte le mappature precedentemente realizzate, per stabilire come era stato mappato un pacchetto con lo stesso header in precendenza, o se si tratta di un replay ad un pacchetto già mappato. Se si realizza un match, viene usata la stessa mappatura. Qualora tutto questo non si verifichi, il pacchetto viene filtrato attraverso le regole, a partire dalla più specifica sino a quella più generica; qualora ci siano più regole che comportino un trattamento del pacchetto, i loro range vengono combinati in un range cumulativo.

Una volta creata la mappatura, se essa è riferita ad un raggio di indirizzi, allora di default viene impiegato quello meno usato al momento, ma, come si è visto, è possibile cambiare politica. Se l'indirizzo mappato risulta già impiegato in una connessione, il NAT opera una traslazione di porta o manda un ICMP di type id per consentire la comunicazione.

Qualora nessuna regola realizzi un match col pacchetto, si crea un null mapping, il pacchetto viene lasciato inalterato. Un null mapping forzato per una classe di indirizzi, (solitamente la intranet), si può ottenere ad esempio con:

```
iptables -t nat -A POSTROUTING -s 1.2.3.0/255.255.255.0
                                     -j SNAT --to-source 1.2.3.0
```

Nel caso un pacchetto sia generato dal localhost e la sua mappatura sia stata manipolata, il source address del suo header diviene quello dell'interfaccia d'uscita, e viene realizzato un Reverse NAT sui pacchetti di risposta.

3.8 Il target TCPMSS

Esiste una ulteriore politica, oltre a quelle già esposte, utilizzabile all'interno delle tre catene di regole di filtraggio e delle chain di NAT: si tratta del TCPMSS. Tale target muta il valore dell'MSS dei TCP SYN packet, consentendo in tal modo di controllare la dimensione massima dei pacchetti delle varie connessioni.

Solitamente si desidera come limite il valore dell'MTU meno 40. In questo modo si evita che alcuni firewall malconfigurati possano bloccare le nostre connessioni quando vengano richiesti dei flussi dati piuttosto consistenti. Configurare correttamente l'MSS è fondamentale per poter utilizzare Linux come firewall che effettui masquerating sulla maggior parte delle linee ADSL in Italia. In caso contrario le macchine dietro al firewall avrebbero difficoltà nel comunicare con molte entità presenti su internet.

Il comando è semplice:

```
iptables -A FORWARD -p tcp --tcp-flags SYN,RST SYN
                                   -j TCPMSS --clamp-mss-to-pmtu
```

Oltre al FORWARD ed OUPUT chain, il target TCPMSS può essere utilizzato all'interno della catena di OUTPUT delle regole di NAT.

3.9 MANGLE ed “iptables -t mangle”

Poco utilizzato, ma in effetti parecchio utile è il packet mangling, ossia la

facoltà di modificare vari bit di configurazione dell'header dei pacchetti, fra cui TOS, DSCP, ECN ecc. Ciò può essere fatto nella catena del PREROUTING od in quella di POSTROUTING/OUTPUT.

Per attivare il packet mangling occorre caricare questi moduli .

```
modprobe ipt_TOS
modprobe ipt_mark
modprobe ipt_tos
modprobe ipt_ecn
modprobe ipt_dscp
modprobe ipt_ttl
modprobe ipt_conntrack
```

Di cosa sia il TOS si è già parlato esaurientemente, vediamo ora un esempio.

```
iptables -t mangle -A PREROUTING -p (protocol tcp/udp/icmp)
-s (source address) --source-port (port) -d (destination address)
                    --destination-port (port) -j TOS --set-tos 4
```

Come si può intuire la sintassi è identica a quella usata per il NAT, la politica viene indicata con **-j TOS** e l'opzione **--set-tos** seguita da un valore numerico (che indica l'8-bit Type of Service field nell'header del pacchetto) indica quale TOS debba essere settato secondo la seguente tabella:

DEC	HEX	TOS
0	(0x00)	Normal-Service
2	(0x02)	Minimize-Cost
4	(0x04)	Maximize-Reliability
8	(0x08)	Maximize-Throughput
16	(0x10)	Minimize-Delay

È necessario specificare sempre il protocollo su cui si intende operare, qualora si tratti del tcp, è possibile anche usare l'opzione **--syn**. Configurare propriamente il TOS è utile per ottimizzare i flussi sulle reti con una banda passante molto ridotta, oppure sottoposte ad un traffico piuttosto intenso.

La configurazione tipica è la seguente:

```
iptables -t mangle -A PREROUTING -p tcp -m tcp --dport 20
                               -j TOS --set-tos 0x08 # (ftp-data)
iptables -t mangle -A PREROUTING -p tcp -m tcp --dport 21
                                    -j TOS --set-tos 0x10 # (ftp)
iptables -t mangle -A PREROUTING -p tcp -m tcp --dport 22
                                     -j TOS --set-tos 0x10 #(ssh)
iptables -t mangle -A PREROUTING -p tcp -m tcp --dport 23
                                  -j TOS --set-tos 0x10 #(telnet)
iptables -t mangle -A PREROUTING -p tcp -m tcp --dport 25
                                    -j TOS --set-tos 0x02 #(smtp)
iptables -t mangle -A PREROUTING -p tcp -m tcp --dport 53
                                    -j TOS --set-tos 0x04 #(bind)
iptables -t mangle -A PREROUTING -p tcp -m tcp --dport 80
                                     -j TOS --set-tos 0x10 #(www)
iptables -t mangle -A PREROUTING -p tcp -m tcp --dport 110
                                    -j TOS --set-tos 0x04 #(pop3)
iptables -t mangle -A PREROUTING -p udp -m udp --dport 110
                                    -j TOS --set-tos 0x04 #(pop3)
iptables -t mangle -A PREROUTING -p tcp -m tcp --dport 143
                                   -j TOS --set-tos 0x04 #(imap2)
iptables -t mangle -A PREROUTING -p udp -m udp --dport 143
                                   -j TOS --set-tos 0x04 #(imap2)
iptables -t mangle -A PREROUTING -p tcp -m tcp --dport 220
                                   -j TOS --set-tos 0x04 #(imap3)
iptables -t mangle -A PREROUTING -p udp -m udp --dport 220
                                   -j TOS --set-tos 0x04 #(imap3)
iptables -t mangle -A PREROUTING -p tcp -m tcp --dport 993
                                   -j TOS --set-tos 0x04 #(imaps)
iptables -t mangle -A PREROUTING -p tcp -m tcp --dport 995
                                   -j TOS --set-tos 0x04 #(pop3s)
```

Servizi come telnet (23), ssh (22), connessione ftp (21) e www (80) richiedono il Minimize-Delay, mentre il data transfer dell'ftp, indubbiamente, funziona meglio

se il TOS è configurato per il Maximize-Troughput.

Per l'SMTP il Minimize-Cost è una soluzione accettabile nella maggior parte dei casi, anche se taluni preferiscono usare il Maximize-Reliability od addirittura il Maximize-Troughput; dipende dal tipo di servizio agli utenti che ci si trova a dover garantire. Servizi come bind, pop ed imap, infine, devone offrire una risposta pronta alle richieste di connessione e transferimento dati, per cui è suggeribile il Maximize-Reliability.

Una funzione simile a quella del TOS è svolta dai 6 bit del DSCP, il cui utilizzo, appunto, dovrebbe sostituirsi nel futuro a quello del Type of Services.

Come si può notare la sintassi è sempre la medesima

```
iptables -t mangle -A PREROUTING -p (protocol tcp|udp|icmp)
                          -s (source address) --source-port (port)
                  -d (destination address) --destination-port (port)
                  -j DSCP --set-dscp (valore decimale o esadecimale)
                           --set-dscp-class (DiffServ class value)
```

Per una lista completa devi valori configurabile per il DSCP si consulti l'RFC2474. Il DiffServ class value può essere EF, BE, oppure qualsiasi della classi CSxx e AFxx.

Con il protocollo tcp, è possibile far ricorso all'opzione **--syn**.

E' noto che parecchi firewall con network layer non aggiornato non sono in grado di trattare in maniera corretta i pacchetti TCP/IP con l'Explicit Congestion Notification impostata, cosí che si limitano a effettuare il DROP dei pacchetti di risposta.

Linux, al contrario, è stato uno dei primi ad includere il supporto all'ECN. Purtroppo, sino a che non tutti i firewall saranno aggiornati, può capitare di dover togliere il bit di ECN dai pacchetti IP provenienti dalla nostra intranet, così che non ci siano mai problemi per la navigazione su Internet. Ciò deve essere fatto, ovviamente, col packet mangling.

```
iptables -t mangle -A PREROUTING -p tcp
                          -s (source address) --source-port (port)
                  -d (destination address) --destination-port (port)
                                            -j ECN --ecn-tcp-remove
```

Come è facile intuire, occorre indicare esplicitamente che si vuole operare solo sul protocollo TCP, ed è possibile usare l'opzione **--syn**. Per impostare un filtro

ancor più accurato si può agevolmente far ricorso al match sul modulo ecn.

```
iptables -t mangle -A POSTROUTING
                                        -m ecn ! --ecn-tcp-cwr -p tcp
                               -s (source address) --source-port (port)
                       -d (destination address) --destination-port (port)
                                                   -j ECN --ecn-tcp-remove
```

Elimina il bit di ECN ai pacchetti in uscita dal firewall, eccetto quello che effettuino il match sul bit CWR dell'header TCP. Un'altra potenzialità del packet mangling che normalmente viene ignorata consiste nel poter associare ad una serie di pacchetti un valore di mark.

```
iptables -t mangle -A PREROUTING -p (protocol tcp|udp|icmp)
                               -s (source address) --source-port (port)
                       -d (destination address) --destination-port (port)
                                                -j MARK --set-mark (valore)
```

Laddove il valore di mark può essere espresso come decimale, od esadecimale (il kernel lo vedrà comunque sempre come un esadecimale).

L'utilizzo più tipico del mark consiste nel creare nel PREROUTING delle macro categorie di pacchetti, che da quel punto in poi saranno appunto identificabili tramite la marcatura inserita nell'header. Naturalmente quando i pacchetti arrivano alle tre catene di regole per il filtraggio porteranno con sé il mark fissato. In questo modo possono essere raggruppati sono una unica identificazione pacchetti diversi per protocollo, funzione, indirizzo di provenienza e destinazione, per poi poterli trattare con un'unica regola in cui si espliciti la ricerca del mark con

```
-m mark --mark (value)
```

Ciò si rivela particolarmente utili qualora si desideri razionalizzare la lista delle regole. Il mark può inoltre venire usato per impostare il QoS utilizzando come il modulo del “firewall based classifier“ .

3.10 LOG ed ULOG

Per abilitare la funzione di logging occorre caricare l'apposito modulo:

```
modprobe ipt_LOG
```

Si tratta di un'estensione delle politiche proprie di tutti i chain, e si indica con **-j LOG**. Sono possibili delle opzioni completive,

--log-level seguito da un numero da 0 a 7, e da un nome **debug**, **info**, **notice**, **warning**, **err**, **crit**, **alert**, **emerg**, corrispondente alla priority da associare al relativo messaggio di log.
--log-prefix seguito da una stringa lunga sino a 14 caratteri, consente di dare un nome al log per poterlo identificare più velocemente fra gli altri.
--log-tcp-sequence logga i tcp sequence numbers
--log-tcp-options logga le options dall'header dei pacchetti TCP
--log-ip-options logga le options dall'header dei pacchetti IP

I log vengono generati tramite delle semplici chiamate **printk()**, per cui sono immediatamente visualizzabili tramite il tradizionale comando dmesg, e sono associati dal demone klogd ai segnali di log di sistema inerenti la facility kern, mentre la priority, come visto è configurabile.
Se il file **/etc/syslog.conf** contiene una linea del tipo:

```
kern.*;kern.none;kern.crit              /var/log/kernel
```

il demone syslogd si occupa di scrivere su file tali messaggi.
La posizione più utile per il monitoraggio del traffico di un firewall, comunque, è il PREROUTING del packet mangling, in modo tale da operare sulla campionatura di tutti i pacchetti che attraversano la macchina.
Spesso conviene usare l'estensione limit per ottenere del log campionario.

```
iptables -t mangle -A PREROUTING -m limit --limit 10000
                                     -p tcp --destination-port 25
                                             -j LOG --log-level 3
                           --log-prefix maillog --log-tcp-options
```

Logga le connessioni per il servizio di posta elettronica.
Il kernel registrerà tutti i pacchetti che passino attraverso il prerouting, e li assocerà ad una chiamata **printk()**, (visualizzabile com dmesg). L'output sarà simile al seguente:

```
maillogIN=eth0 OUT= MAC=00:00:00:00:00:00:00:00:00:00:00:00:08:00
SRC=192.167.206.129 DST=212.34.32.15 LEN=60 TOS=0x10 PREC=0x00 TTL=64
ID=0 DF PROTO=TCP SPT=1503 DPT=25 WINDOW=32304 RES=0x03 SYN
URGP=0 OPT (02043F180402080A0044A87D0000000001030300)
```

A seconda del livello di log scelto, per il kernel cambierà il tipo di notifica cui associare il log per il syslogd, consentendo di registrare i messaggi di notifica nel log standard di sistema.

L'utilizzo del syslog, per quanto rapido, non è sempre la soluzione più congeniale ai bisogni che si possono verificare quando si amministra una rete con un traffico molto variegato. Molto più comodo è deviare il log in user space, per poterlo quindi intercettare con un listener e scriverne i messaggi su un file specifico od ancor meglio su un data base. Per fare questo il Netfilter di Linux mette a disposizione il target ULOG. Il funzionamento è molto semplice. I messaggio di log vengono girati all'user space utilizzando dei netlink multicast socket, i quali sono intercettati da un demone, solitamente chiamato ulogd, in ascolto.

In primo luogo è necessario caricare l'apposito modulo del kernel.

```
modprobe ipt_ULOG
```

La sintassi è molto simile a quella del target LOG, ma i parametri per la configurazione granulare

```
iptables -t mangle -A PREROUTING -p (protocol tcp/udp/icmp)
-s (source address) --source-port (port) -d (destination address)
 --destination-port (port) -j ULOG --ulog-nlgroup (netlink group)
                    --ulog-cprange (dimensione) --ulog-qthreshold
                                   --ulog-prefix (prefisso del log)
```

--ulog-nlgroup (nlgroup)	indica il gruppo NETLINK usato per il log
--ulog-cprange <dimensione>	indica quanti byte di ciascun pacchetto debbano essere passati all'user space
--ulog-qthreshold Threshold	della in-kernel queue
--ulog-prefix (prefisso)	indica con quali prefisso il log venga comunicato al demone in ascolto.

Abbiamo accennato al fatto che solitamente per intercettare il log in user space viene utilizzato il demone ulogd. Tale software può essere scaricato dall'url:
http: //www.gnumonks.org/projects/ulogd

La configurazione dell'ulogd, ancorché semplice, è estremamente flessibile. In base ai parametri impostati nei filtri con target ULOG è possibile far sí che il demone scriva i messaggi a lui rivolti in differenti file, oppure in varie tabelle all'interno d'un database MySQL.

3.11 Bastano poche regole per un firewall sicuro

A questo punto si è finalmente in grado di configurare un vero e proprio firewall. L'esempio qui proposto si adatta ad una piccola intranet, che esca su Internet con un solo indirizzo IP pubblico, senza che da Internet vi sia la possibilità di accedere a nessun servizio erogato all'interno della intranet o dal firewall stesso. Si tratta di una situazione ormai piuttosto comune. Si pensi a chi voglia consentire a più di un solo computer di uscire su Internet tramite la propria connessione ADSL, e tuttavia desideri sentirsi protetto da eventuali attacchi. Dal momento che la maggior parte delle connessioni di codesto genere hanno un indirizzo assegnato dinamicamente è necessario indicare espressamente nella linea di comando le interfacce di rete in luogo degli indirizzi IP.

```
# Local Interface
# This is the interface that is your link to the world

LOCALIF="eth0"

# Internal Interface
# This is the interface for your local network
# NOTE: INTERNALNET is a *network* address. All host bits should be 0

INTERNALNET="192.168.200.0/255.255.255.0"

# External Interface
# For ADSL connections which use ppp over ethernet, it usually should be ppp0

EXTERNALIF="ppp0"

# External NetWork (will be unused)
REMOTENET="0/0"

# Set the location of iptables
IPTABLES="/sbin/iptables"

# -- Flush everything, start from scratch -

echo -n "Flushing rulesets.."

# Rules from the filter chains

$IPTABLES -F
echo -n "."
```

```
# Rules from the mangle chains
$IPTABLES -t mangle -F
echo -n "."

# Rules from the nat chains
$IPTABLES -t nat -F
echo -n "."

echo "Done!"

# Set up filter chains
echo -n "Filter chains.."

# Set filter chains default policy to DROP
$IPTABLES -p INPUT DROP
$IPTABLES -p FORWARD DROP
$IPTABLES -p OUTPUT DROP
echo -n "..."

# First of all, drop unclean packages
$IPTABLES -A INPUT -m unclean -j DROP
echo -n "."

# Then drop spoofed packets
$IPTABLES -A INPUT -s $INTERNALNET -i $EXTERNALIF -j DROP
echo -n "."

# Allow the firewall con connect to 0/0, and to accept ssh connections
# from out LAN

$IPTABLES -A INPUT -s $INTERNALNET -i $LOCALIF
                                    -p tcp -m tcp --dport 22 -j ACCEPT
$IPTABLES -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
echo -n "..."

# Protect the firewall from icmp DoS
$IPTABLES -A INPUT -p icmp -m icmp --icmp-type 8
                                  -m length --length 128:65535 -j DROP
echo -n "."

# this is mandatory to route an ethernet LAN to a PPPOE connection
$IPTABLES -A FORWARD -p tcp
          -m tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu
echo -n "."
```

```
# Set up the forward chain, to allow connection from out LAN to 0/0,
# refusing 0/0 connections to our LAN
$IPTABLES -A FORWARD -i $EXTERNALIF
                        -m state --state RELATED,ESTABLISHED -j ACCEPT
$IPTABLES -A FORWARD -i $INTERNALIF
                    -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
echo -n "..""

# Protect out LAN from icmp DoS
$IPTABLES -A FORWARD -p icmp
            -m icmp --icmp-type 8 -m length --length 128:65535 -j DROP
echo -n "."

# Do not route multicast packets
$IPTABLES -A FORWARD -m pkttype --pkt-type multicast -j DROP
echo -n "."

Echo "Done!"

# Set up nat chains
echo -n "Nat chains.."

# Set up Masquerading
$IPTABLES -t nat -I POSTROUTING
                          -s $INTERNALNET -o $EXTERNALIF -j MASQUERADE
echo -n "."

Echo "Done!"

# Set up mangle chains
echo -n "Mangle chains.."

# Log to ulogd icmp DoS
 $IPTABLES -t mangle -A PREROUTING -p icmp -m icmp --icmp-type 8
                                  -m length --length 128:65535 -j ULOG
echo -n "."

# This section manipulates the Type Of Service (TOS) bits of the
# packet.

# Minimize-Delay
$IPTABLES -t mangle -A PREROUTING -p tcp -m tcp
               -m multiport --dports 21,22,23,80 -j TOS --set-tos 0x10
echo -n "."


# Maximize-Throughput
```

```
$IPTABLES -t mangle -A PREROUTING -p tcp -m tcp
                    -m multiport --dports 20,8080 -j TOS --set-tos 0x08
echo -n "."

# Maximize-Reliability
$IPTABLES -t mangle -A PREROUTING -p tcp -m tcp --dport 53
                                                    -j TOS --set-tos 0x04
echo -n "."
echo "Done!"
```

3.12 Salvare e ricaricare le regole

Mantenere una script come quella proposta poco sopra potrebbe divenire nel tempo dispendioso e poco pratico. **iptables** ha dei comandi propri per salvare ed operare il restore delle regole, grazie ai quali delle script di configurazione divengono superflue.

Una volta configurato il nostro firewall, le regole impostate possono venire salvate tramite il comando:

```
iptables-save > file_di_salvataggio
```

Si noti che è necessario ridirigere espressamente l'output all'interno di un file, il cui contenuto, nel caso del firewall che abbiamo configurato sin qui passo passo, sarà simile a questo (si consideri **eth0** come interfaccia di input per la rete locale ed **eth1** per la extranet; il terminale d'amministrazione avrà indirizzo 192.168.200.189) .

Il formato è molto semplice, così che il file può essere tranquillamente editato anche a mano. Dopo un reboot o più semplicemente un flush delle regole è possibile ripristinare la configurazione primigenia del firewall tramite:

```
iptables-restore < file_di_salvataggio
```

È buona norma, per i firewall di produzione, far caricare le regole immediatamente al boot, inserendo di comando di restore nel file **rc.local**, oppure negli script ad hoc per la singola distribuzione.

Allegato redazionale a Linux&C. 34 - Ottobre 2003

Editore: Piscopo Editore s.r.l
Via di Villa Sacchetti, 11
00197 Roma

Direttore Responsabile: Annabella Faustini
a.faustini@oltrelinux.com

Direttore Editoriale: Patrizio Tassone
p.tassone@oltrelinux.com

Testi a cura di: Luigi Genoni
l.genoni@oltrelinux.com

Editing e Redazione: Edizioni Vinco,
Strada Casa Bianca, 52
43100 Parma (PR)

Registrazione al Tribunale di Roma: 332/99 del 17.07.19999